Deep Interactive Text Prediction and Quality Estimation in Translation Interfaces

Christopher M. Hokamp
B.A., M.A.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University
School of Computing

Supervisors:
Prof. Qun Liu
Prof. Josef van Genabith

September 2018

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Chris Hokamp

ID No.: 13211891

Date: 12 September 2018

Acknowledgements

Over the last five years, I have been incredibly lucky to learn from and collaborate with many brilliant researchers.

I am particularly indebted to my supervisor Qun Liu. From my first day in the lab until the final throes of revision, he was a constant source of guidance, wisdom, and support. His feedback and insights during innumerable conversations helped my research to take form, and his inspirational work ethic helped me to push through some of the more difficult moments.

Josef van Genabith originally inspired me to begin my PhD, and his positive energy was a motivating force throughout my time in the EXPERT project.

Rada Mihalcea's graduate course in NLP at the University of North Texas made me fall in love with this field. Rada is a brilliant teacher and mentor, and our conversations encouraged me to pursue a career in research.

My examiners Mikel Forcada and Jennifer Foster provided detailed constructive feedback on this work, which helped immensely.

A big Irish cheers to all of my collaborators, and to the many friends from DCU, Unbabel, and the EXPERT project. Impromptu conversations and brainstorms frequently resulted in insights, shared knowledge, useful hacks, or actual publications. I feel really lucky to have met so many fun and dynamic people, who also happen to be crazy smart.

We had a blast during the four-year run of the DCU machine learning group — this was one of my favorite parts of my time at DCU. I would like to thank everyone who helped make our group an great resource for learning, especially my amazing labmate and neighbor Piyush Arora, Joachim Wagner, and Henry Elder.

I am eternally grateful to the EXPERT project, and to the ADAPT Centre. Their support for my research, and the nice environment in our lab at DCU made this work possible.

Finally, I would like to thank all of my family and friends, especially Merve and Maya, my parents Mark and Barbara, and Mehmet and Ayşegül, for their love, support, and patience.

Contents

Deep Interactive Text Prediction and Quality Estimation in Translation Interfaces

Christopher Mark Hokamp

Abstract

The output of automatic translation systems is usually destined for human consumption. In most cases, translators use machine translation (MT) as the first step in the process of creating a fluent translation in a target language given a text in a source language. However, there are many possible ways for translators to interact with MT. The goal of this thesis is to investigate new interactive designs and interfaces for translation.

In the first part of the thesis, we present pilot studies which investigate aspects of the interactive translation process, building upon insights from Human-Computer Interaction (HCI) and Translation Studies. We developed HandyCAT, an open-source platform for translation process research, which was used to conduct two user studies: an investigation into interactive machine translation and evaluation of a novel component for post-editing.

We then propose new models for quality estimation (QE) of MT, and new models for estimating the confidence of prefix-based neural interactive MT (IMT) systems. We present a series of experiments using neural sequence models for QE and IMT. We focus upon token-level QE models, which can be used as standalone components or integrated into post-editing pipelines, guiding users in selecting phrases to edit. We introduce a strong recurrent baseline for neural QE, and show how state of the art automatic post-editing (APE) models can be re-purposed for word-level QE. We also propose an auxiliary confidence model, which can be attached to (I)-MT systems to use the model's internal state to estimate confidence about the model's predictions.

The third part of the thesis introduces lexically constrained decoding using grid beam search (GBS), a means of expanding prefix-based interactive translation to general lexical constraints. By integrating lexically constrained decoding with word-level QE, we then suggest a novel interactive design for translation interfaces, and test our hypotheses using simulated editing. The final section focuses upon designing an interface for interactive post-editing, incorporating both GBS and QE. We design components which introduce a new way of interacting with translation models, and test these components in a user-study.

List of Figures

List of Tables

List of Acronyms

AI artificial intelligence.

APE automatic post-editing.

API application programming interface.

CAT computer-aided translation.

CCD component-centric design.

CD constrained decoding.

CE confidence estimation.

CL computational linguistics.

DL deep learning.

DOM document object model.

DSL domain-specific language.

EOS end-of-sentence.

GBS grid beam search.

GPU Graphical Processing Unit.

GRU gated recurrent unit.

HCI human-computer interaction.

HTER human translation edit rate.

I-NMT interactive neural machine translation.

IMT interactive machine translation.

IPE interactive post-editing.

IPT interactive-predictive translation.

ITP interactive translation prediction.

JSON Javascript object notation.

KSMR keystroke-mouse ratio.

KSR keystroke ratio.

LSP language service provider.

LSTM long-short term memory.

MEMD maximum entropy/minimum divergence.

MERT minimum error rate training.

ML machine learning.

MLP multi-layer perceptron.

MT machine translation.

NER named entity recognition.

NLP natural language processing.

NMT neural machine translation.

NN neural network.

OOV out-of-vocabulary.

PB-SMT phrase-based statistical machine translation.

PE post-editing.

PMI pointwise mutual information.

POS part-of-speech.

PRIMT pick-revise machine translation.

QE quality estimation.

RNN recurrent neural network.

SCFG synchronous context-free grammar.

seq2seq sequence-to-sequence.

SMT statistical machine translation.

SRN simple recurrent network.

SVM support vector machine.

TER translation error rate.

TM translation memory.

TSV tab-separated value.

UI user interface.

WMT workshop on machine translation.

Chapter 1

Introduction

The output of automatic translation systems is destined for human consumption. Before the advent of modern machine translation systems, all translations were produced by humans with knowledge of the particular pair of source language and target language. Although human translators produce high-quality results, there is an upper bound in terms of the throughput that a translator can achieve, and there will always be a finite number of expert translators for any given language pair. The explosion of multilingual content in the internet age, fueled by the globalization of modern markets, has driven the need to increase the speed of translation services by several orders of magnitude. Over the last three decades, automatic translation systems with the necessary efficiency profile have been invented and deployed at global scale, but the quality of machine translation cannot be relied upon in many real-world scenarios, and humans are likely to remain in the loop for the foreseeable future.

In spite of its imperfections, machine translation (MT) has nevertheless become a central part of translation workflows, because humans can intervene to control the final output, or guide MT systems towards producing good translations. As MT has become commonplace, fast, affordable, and accessible, new ways of interacting with translation systems have begun to emerge. As a result, the translator's task has differentiated into several fine grained sub-tasks, each with its own intricacies and necessary skills. One of the goals of this thesis is to investigate these different modes of interaction between humans and automatic translation systems, by building tools that can be used to test diverse interactive possibilities in the context of computer-aided translation (CAT).

A second goal is to improve models of translation quality, and models which support interactive translation, by exploring new designs, training paradigms, and algorithms. Recent work in deep learning, translation quality estimation, and interactive machine translation has created new possibilities for integration of natural language processing tools into user interfaces for translation; however, the quality of the output of these models remains the major barrier to their widespread adoption and deep integration into all translation tools. We thus explore ways to improve models of translation quality, and explore new ways of guiding and controlling the output of MT models as translations are being generated.

With these two main goal in mind, our high-level research objective becomes the discovery and evaluation of models, algorithms, and interfaces that can allow translators to make better use of MT. We take an integrative approach, investigating all parts of the translation pipeline, from developing and evaluating new MT and quality estimation models, to creating new search algorithms which expand the possibilities for guiding the output of these models, to exposing translation systems as real-time predictors, to designing and testing the interfaces which allow interaction between humans and translation models.

On the machine learning side, we contribute new models for the word level *quality estimation* of MT, *automatic post-editing* (APE), and *interactive machine translation* (IMT), as well as intrinsic methods for estimating the confidence of a translation model about its predictions. We also propose new ways of searching for optimal outputs from translation models, allowing translators to provide any part(s) of a translation hypothesis, and then query an MT system for the other parts. The latter case, which we refer to as *lexically constrained decoding*, generalizes the paradigm that has traditionally been called IMT, or Interactive-Predictive MT (Foster, 2002; Green, 2014), allowing MT models to produce outputs subject to multiple externally-specified lexical constraints. We show that this technique can be useful in both interactive and fully-automatic translation scenarios, and demonstrate several possible integration points into real-world translation workflows. In the final part of the thesis, we combine lexically constrained decoding with QE models to enable a new interactive modality for CAT, which we call *interactive post-editing* ().

Because our end objective is to create models and systems that can be integrated into

real translation workflows, we also design a user interface for Computer Aided Translation that enables proof-of-concept implementations allowing interaction between humans and translation models. We conduct several user studies with our open-source interface, testing new interactive scenarios and graphical component designs with real translators.

1.1 Research Objectives

Given the high level goals laid out above, we can now be more specific, and enumerate three high-level research questions which guide this work:

> **RQ1**: *How can we test the integration of novel translation technologies within computer-aided translation interfaces?*
>
> **RQ2**: *How can we improve the performance of models which estimate MT quality at the word level?*
>
> **RQ3**: *Can we extend interactive MT beyond prefix decoding, and find a general framework which allows the specification of any kind of lexical constraints in MT output?*

Throughout the thesis, we will try to find answers to each of these questions, and we will also look for synergies between them, working toward a unified view of CAT integrated with MT and models of translation quality.

1.2 Structure

The thesis has six core chapters, which are organized as follows:

Chapter 2 reviews the background of this work, and gives the historical and academic context for the research conducted in this thesis. We give an overview of related work for each of the chapters; however, individual chapters may also discuss related work, where more detail on a specific topic is required. We try not to repeat content, so where a topic is discussed in more depth in a particular chapter, we refer the reader to that portion of the thesis.

Chapter 3 presents HandyCAT, our platform for translation process research. HandyCAT was designed as a flexible, web-based interface, particularly well-suited to the implementation and testing of new graphical components for translation. We discuss the design philosophy and the practical implementation details of the interface, and propose an general object-oriented design framework for CAT interfaces, which tries to extract some core abstractions that all CAT tools have in common. The second part of this chapter presents two pilot studies conducted in collaboration with translation studies researchers, which investigated interactive autocompletion components, and components designed for post-editing, building upon insights from human-computer interaction (HCI) and translation studies research.

Chapter 4 proposes new models for the quality estimation (QE) of MT, and presents a framework for extracting features for QE models. We then discuss a series of experiments using neural sequence models. This chapter is focused upon token-level QE models, which can be used as standalone components or integrated into post-editing pipelines, guiding users in selecting phrases to edit, an idea we implement and test in chapter 7. Chapter 4 concludes with a set of experiments that evaluate models designed to perform both word-level QE and automatic post-editing (APE) simultaneously, introducing extensions to state-of-the-art neural APE models which achieve high-quality results in both QE and APE.

In **Chapter 5**, we present models and experiments evaluating usecases for confidence prediction within prefix-based neural interactive machine translation. We show how an neural machine translation (NMT) system can be easily converted into an interactive-NMT (I-NMT) system, an approach which several research groups have recently capitalized upon. This chapter is focused upon the search for ways to maximize the utility of I-NMT model output, by truncating predictions and delegating to the translator when system confidence is low.

Chapter 6 zooms out from MT-specific models, and introduces *lexically constrained decoding*, a general algorithm for forcing arbitrary lexical constraints to appear in the output of a large class of models which generate natural language conditioned upon some input. We show that a general algorithm exists that can decode outputs that are guaranteed

to contain lexical constraints, and confirm the utility of our algorithm with experiments on simulated interactive post-editing, and automatic domain adaptation of MT.

Chapter 7 integrates word-level QE and lexically constrained decoding into our CAT interface, presenting a proof-of-concept design for *interactive post-editing* (IPE). Word level QE is used to guide translators toward the portions of the MT output which are likely to need editing, while lexically constrained decoding is used to generate new translations which preserve any editing that a translator has already done. To the best of our knowledge, our design is completely novel, and is the first tool to combine interactive word-level QE, and an interactive MT model that can produce translations given user constraints, into a real time CAT interface that can actually be used for translation. We design and implement the IPE interface, and conduct a preliminary test of the tool in a very small user study with professional translators.

1.3 Contributions

In a nutshell, the main contributions of this thesis are:

- Design and evaluation of new interfaces for interactive computer-aided translation
- New models for word-level quality estimation and automatic post-editing
- Evaluations of novel methods of modeling confidence in interactive machine translation
- A framework for inserting hard lexical constraints into the outputs of sequence-to-sequence models

1.4 Publications

Here we list our peer-reviewed publications which are directly related to the contents of this thesis, organized by their high-level topical categories:

1.4.1 Computer-Aided Translation

- Dave Lewis, Qun Liu, Leroy Finn, Chris Hokamp, Felix Sasaki, and David Filip. Open, web-based internationalization and localization tools. *Translation Spaces, vol III*, 3(1):99–132, 2014. doi: doi:10.1075/ts.3.05lew

- Chris Hokamp. Leveraging NLP technologies and linked open data to create better CAT tools. *Localisation Focus - The International Journal of Localisation*, 14, 2015. ISSN 1649-2358

- Chris Hokamp and Qun Liu. Handycat - an open-source platform for CAT tool research. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation, EAMT 2015, Antalya, Turkey, May 11 - 13, 2015*, 2015. URL `https://aclanthology.info/papers/W15-4934/w15-4934`

1.4.2 Translation Quality Estimation

- Chris Hokamp, Iacer Calixto, Joachim Wagner, and Jian Zhang. Target-centric features for translation quality estimation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 329–334, 2014. URL `http://aclweb.org/anthology/W/W14/W14-3341.pdf`

- Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W15-3001`

- Varvara Logacheva, Chris Hokamp, and Lucia Specia. Data enhancement and selection strategies for the word-level quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 311–316, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W15-3039`

- Varvara Logacheva, Chris Hokamp, and Lucia Specia. MARMOT: A toolkit for translation quality estimation at the word level. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC*, Por-

torož, Slovenia, 2016. URL http://www.lrec-conf.org/proceedings/lrec2016/summaries/1054.html

- André F. T. Martins, Ramón Astudillo, Chris Hokamp, and Fabio Kepler. Unbabel's participation in the WMT16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*, pages 806–811, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2387

- André Martins, Marcin Junczys-Dowmunt, Fabio Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. Pushing the limits of translation quality estimation. *Transactions of the Association for Computational Linguistics*, 5:205–218, 2017. URL https://transacl.org/ojs/index.php/tacl/article/view/1113

- Chris Hokamp. Ensembling factored neural machine translation models for automatic post-editing and quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 647–654. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/W17-4775

1.4.3 Machine Translation and Machine Learning

- Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-1141

1.4.4 Other Publications

The following peer-reviewed publications were published during the course of this thesis, but their content is not directly related to the work presented here:

- Longyue Wang, Chris Hokamp, Tsuyoshi Okita, Xiaojun Zhang, and Qun Liu. The DCU discourse parser for connective, argument identification and explicit sense classification. In *Proceedings of the 19th Conference on Computational Natural Lan-*

guage Learning: Shared Task, CoNLL 2015, Beijing, China, July 30-31, 2015, pages 89–94, 2015. URL http://aclweb.org/anthology/K/K15/K15-2014.pdf

- Mahmoud Azab, Chris Hokamp, and Rada Mihalcea. Using word semantics to assist english as a second language learners. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 116–120, 2015. URL http://aclweb.org/anthology/N/N15/N15-3024.pdf

- Chris Hokamp and Piyush Arora. Dcu-semaniacs at semeval-2016 task 1: Synthetic paragram embeddings for semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 656–662, 2016. URL http://aclweb.org/anthology/S/S16/S16-1100.pdf

- Peyman Passban, Chris Hokamp, Andy Way, and Qun Liu. Improving phrase-based SMT using cross-granularity embedding similarity. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation, EAMT 2017, Riga, Latvia, May 30 - June 1, 2016*, pages 129–140, 2016. URL https://aclanthology.info/papers/W16-3403/w16-3403

Chapter 2

Background

This chapter introduces the essential background for the work in this thesis, highlighting the connections between CAT tool design, interactive translation, and quality estimation of MT. The topics we consider span the fields of computational linguistics (CL), machine learning (ML), natural language processing (NLP), human-computer interaction (HCI), and translation studies.

In order to organize the discussion, the chapter follows the overall organization of the thesis, beginning by examining human factors in translation, and proceeding to an overview of translation interfaces in general. We then give a brief introduction to deep learning (DL) and neural machine translation (NMT), since recent progress in these areas underpins most of the machine learning work presented in this thesis. We then move to an overview of the algorithms and computational techniques used in interactive machine translation and conclude with an overview of word level quality estimation (QE), and automatic post-editing (APE).

2.1 Translation and Human-Computer Interaction

Translation interfaces provide a means for humans to transform text in one language into text in another language. This transformation is achieved via an intricate series of actions taken by the human user, involving some degree of planning ahead, followed by the actual production of the translation, and possibly finishing with one or more revision passes. From the point of view of human-computer interaction, we may be interested in both the hows and the whys of the unfolding of this process:

Hows

- how do translators use the UI components that enable translation?
- how do translators use machine translation?
- how can we automate parts of the translation process?

Whys

- why do translators perform certain actions in certain orders?
- why are some translation tasks more difficult than others?
- why are some interactions more efficient than others?

The field of translation studies views the translator as the central object of study, attempting to understand the cognitive processes underlying translation, and the core strategies that translators use (Lörscher, 1996; Chesterman and Wagner, 2002).

Computers are obviously ubiquitous in all areas of modern life, and over the last four decades or so, translators have necessarily transitioned to working almost exclusively on computers. Therefore, a subset of translation studies researchers have focused upon studying the interactions between translators and their tools, connecting translation studies with HCI, or translator-computer interaction (TCI) (Kay, 1997; O'Brien, 2012; Bowker and Fisher, 2010).

Automation is the fundamental reason that computers are useful to humans. By automating time-consuming or tedious tasks, computers can help humans to complete a task

more quickly, or with less effort. It follows that the primary measure of the effectiveness of an interface should be the degree to which it enables users to perform a task more quickly and with less effort when compared with alternative methods. In the specific case of translation, we may assume that the translator's goal is to perform the mapping from source to target as easily and quickly as possible, and that the purpose of a translation interface is to help the translator to achieve that goal. Of course, this simplistic view assumes that the quality of the resulting translation will always be above some standard, regardless of how it is produced. In other words, we believe that the translator will not sacrifice the *quality* of the translation in order to complete a task more quickly, an assumption that may or may not be borne out in practice. However, with this assumption in place, we may proceed to define metrics which score translation interfaces by measuring their utility to human translators.

Time-to-completion and ease of completion may be bundled together with the label *efficiency*. Efficiency can be a purely time-based measure, meaning that more efficient approaches lead to faster completion of a translation task, or may incorporate measures of cognitive load and quality of the output into account as well. Because cognitive load is not directly measurable, some researchers have proposed metrics that are assumed to correlate well with cognitive load, such as the number of keystrokes ("technical effort"), or eye movements and fixation duration (Carl, 2010; Koponen et al., 2012; Federico et al., 2012; Martínez-Gómez et al., 2014).

2.1.1 Mixed-Initiative CAT

The mixed-initiative model (Kay, 1997; Horvitz, 1999; Green, 2014) of CAT, introduces an interactive model of the translation process, where the user and the translation system(s) work together to achieve a shared goal, namely to produce a target translation that preserves the meaning of the source text, while also conforming to the linguistic and social conventions of the target language. The degree to which this collaborative effort has succeeded can be measured by comparing the efficiency of the mixed-initiative system with a translator's efficiency without the automatic system.

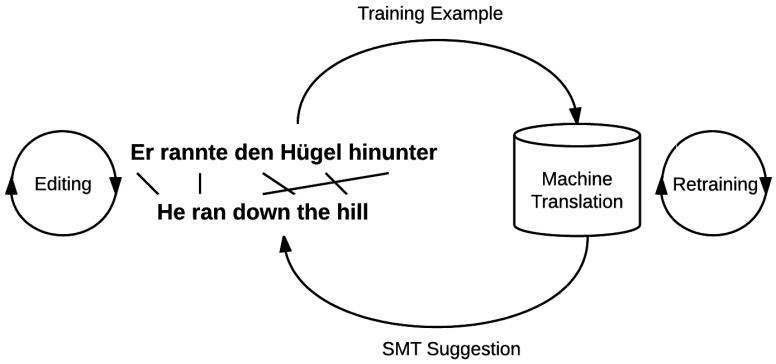


Figure 2.1: Interactive translation workflows, modeled after (Green, 2014). Explicit interaction may occur on the left side of the figure, if the translator is directly giving instructions to the MT system (see section 2.4). Implicit interaction happens when the MT system on the right side retrains on data obtained from the translation interface. Updates to the MT system may happen instantly, or may be delayed.

2.1.2 Modes of Interaction

Interaction between translators and machine translation systems may take place *implicitly* or *explicitly* (Figure 2.1).

Implicit Interaction Implicit interaction refers to cases where users are unaware that an interface is adapting to or learning from their input and cases where users do not need to modify anything about their use of an interface in order for adaptation to occur. In MT research, implicit interaction approaches have focused upon the retraining of MT engines given feedback from translation or post-editing (Ortiz-Martinez, 2011; Martínez-Gómez et al., 2012; Ortiz-Martínez et al., 2010; Mathur et al., 2013; Dara et al., 2014; Denkowski et al., 2014; Blain et al., 2015). In this line of research, the focus is not upon testing new features in a user interface. Instead, the goal of online retraining is to immediately update the model(s) of the translation system with each completed translation. If the translation system successfully learns to create translations which require less effort for the translator to post-edit, the translator can provide feedback in the form of new post-edits more quickly, resulting in a positive feedback loop between translator productivity and MT system quality.

Active learning (Settles, 2010; Mathur et al., 2013; Dara et al., 2014) allows an MT system to actively participate in its own retraining, by selecting the segments of the translation task for which it is least likely to produce an adequate translation. Human trans-

lators are then prompted to provide translations for these "high-entropy[1]" segments first, and the system is then retrained on the new translations, so that the overall quality of the MT system increases more rapidly than it would if the segments to translate were chosen in document order, or in a random order.

Most research on implicit interaction does not actually test the proposed system with human users, because the active learning process can be simulated by using a parallel corpus, where the reference translations can be used as simulated post-edits of the SMT system output. The metrics used to measure the quality of a system's output over the course of training, such as the edit distance between the model's translation and a reference translation are assumed to correlate with the effort required to post-edit a translation, and can be computed automatically (Dara et al., 2014).

Explicit Interaction

Explicit interaction refers to scenarios where users are consciously collaborating with predictive models in order to achieve a shared goal. This means the user is providing input and receiving direct visual feedback on how they are influencing the translation process. Explicit interaction scenarios may be validated by user studies (Barrachina et al., 2009; Casacuberta et al., 2009; Koehn, 2009b; Green et al., 2014a), or by automatically simulating users' actions, making similar assumptions to those mentioned in the implicit scenarios discussed above — see for example (Ueffing and Ney, 2005; Ortiz-Martinez, 2011).

In this thesis, we are primarily interested in explicit interaction. Our motivating scenarios are circumstances where users are aware of the interactive nature of the task, and we wish to design models which aid users in achieving an explicit goal.

2.1.3 CAT Tasks

We focus upon four high-level tasks that may be performed using a translation interface:

- Translation from scratch
- Interactive translation

[1]The term entropy is used here because segments are often selected by choosing the candidates with the maximum perplexity according to the translation systems current language model.

- Post-editing
- Interactive post-editing

The following sections define each of these tasks in turn, while our implementations of components that enable each of these interactive modalities are discussed in chapter 3.

Translation from scratch

In this scenario, translations are composed by typing characters on the keyboard, starting from an empty prompt.

Interactive translation

Interactive translation, often called *interactive machine translation* (IMT), is an paradigm where the translator cooperates with a machine translation system to build a translation. Instead of entering each character of the translation, the translator has the option to allow the machine translation system to auto-complete portions of the translation. Importantly, the output is always under the user's control – in other words, the interface expects an explicit user action before automatically-generated content can be inserted into the translation. Section 2.4 discusses IMT in more detail.

The terms *interactive-predictive translation* (IPT) and *interactive translation prediction* (ITP) have recently become popular, likely because they avoid some of the biases associated with the use of the term IMT as introduced above. In the context of this thesis, these three terms (IMT, IPT, and ITP) have the same meaning — we are interested in scenarios where users interact with an intelligent system to compose translations (Torregrosa et al., 2017; Green, 2014), but we do not make assumptions about the specific mode of interaction until later chapters.

Post-editing

When *post-editing*, a translator begins with a translation hypothesis, and must correct the hypothesis until it is a good translation. The hypothesis may be generated by a machine translation system, or it may be a match from a translation memory, where the source side of the translation memory entry is the same as or similar to the current source segment to be translated.

The post-editing process can lead to significant gains in translator productivity when compared to translation from scratch (Koehn, 2009b; Specia, 2011b; Green et al., 2013; Alabau et al., 2013b). However, post-editing can be a tedious task, both because it is typically done in an interface which is designed for composition (writing new text), not for editing (revising provided text), and because the MT systems integrated into professional CAT tools typically do not take user feedback into consideration, leading to the MT system to make the same mistakes, regardless how often they are corrected (Denkowski et al., 2014; Haffari and Sarkar, 2009; Green, 2014). There is also a direct dependency between the MT system outputs, and the amount of post-editing necessary. It is therefore important to ensure a minimum quality threshold for the MT output, as measured by the average or predicted effort required to post-edit its translations, because a low-quality MT system could make post-editing even slower than translation from scratch.

Interactive post-editing

Some recent work has used the term interactive post-editing (IPE) to label different extensions to the post editing task, providing either new interactions which make post editing more efficient, or new ways of automatically proposing suggestions to help translators to fix errors that may occur anywhere in the translation hypothesis (Alves et al., 2016; do Carmo, 2017). In addition, some new commercial translation tools, such as Lilt[2], provide user-interfaces that are purpose built for post-editing, as well as incorporating support for interaction between translators and MT models. In this thesis, we propose a new version of IPE, which uses word-level quality estimation, and constrained decoding in tandem to guide and assist translators during post editing (see chapter 7).

2.2 Interfaces for Translation

Tools for text-editing were some of the first graphical applications ever designed, and modern versions of text-editing interfaces are doubtless the dominant means by which new textual content is created in all major languages. Certain graphical *affordances* (Gaver, 1991; Greeno, 1994) are present in all graphical text editing tools, and all modern text editors implement a core set of common functionalities, such as highlighting

[2] https://lilt.com/

to edit spans of text, or automatic spell checking,

Although modern text editors such as Microsoft Word can be used to compose translations, they lack special functionality for inspecting, analyzing and interacting with source sequences. The need to *transform* source language text into target language text, as opposed to *creating* text in the target language, means that the translation interfaces are designed for a much narrower and more specialized task. Graphical interfaces designed specifically for translation (CAT tools) provide specialized affordances for translation via components designed to streamline the process of transforming text from one language to another (Hutchins, 1998).

The translation memory (TM) was the original technology motivating the development of specialized CAT tools (Hutchins, 1998; Melby, 1982). At its core, a translation memory is a database of source $\rightarrow$ target translations. CAT interfaces automatically search this database to find source segments which match the current segment being translated. If a matching segment is found, the translator may simply use its corresponding target language translation, instead of translating the source segment from scratch. Using a database match is obviously more expedient than typing an entire translation, and may have the additional benefit of ensuring consistency across documents or translation jobs. The search for matching segments typically also allows for some degree of "fuzziness", allowing translators to reuse partially matching segments which may require some post-editing on the target side.

Surprisingly, research into CAT tools is much more recent than research on fully-automatic machine translation (Hutchins, 1998; Kay, 1997). High-quality automatic translation between different languages is a long standing goal of artificial intelligence, and has been a very active area of investigation since the early 1960s (Bar-Hillel, 1960; Slocum, 1985). However, it took about two decades before researchers began serious efforts to design tools to aid translators in their work. Some digital translation aids were created during the late 1970s, and terminal-based tools were in relatively widespread use by the mid-1980s; translator workbench software for personal computers started being widely used in the early 1990s (Hutchins, 1998). The core components used in translation tools today were identified during the 1980s: translation memories, concordance

engines, and terminology management software, as well as the split-screen interface partitioning, where source segments are shown above or to the left of target segments. An object-oriented view of CAT tools, which tries to capture the key abstractions shared by this class of interfaces, is discussed in more detail in Chapter 3.

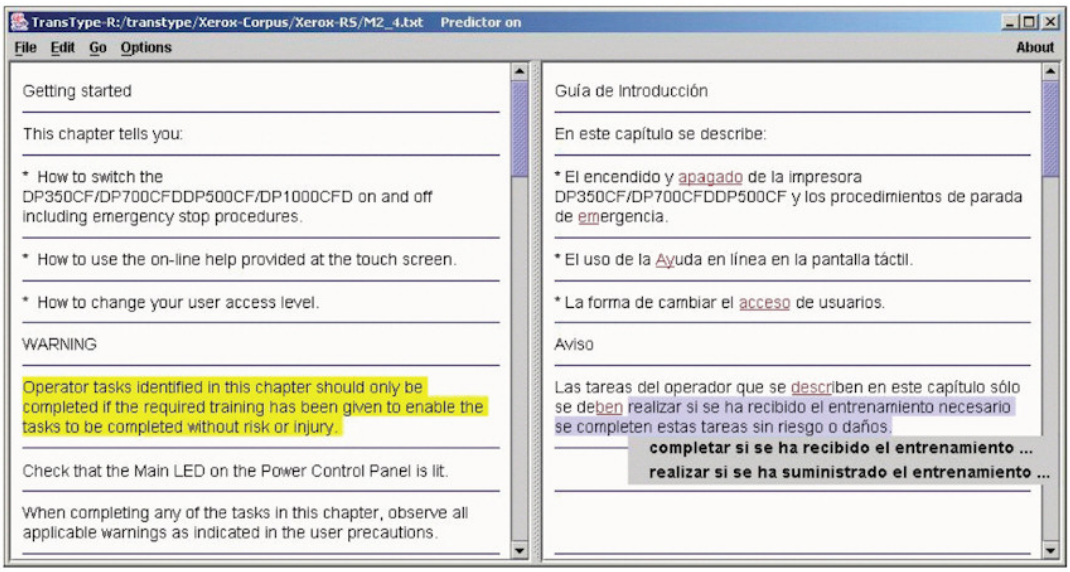
Modern CAT interface research is conducted from two main perspectives. As mentioned in the previous section, researchers in translation studies generally approach CAT research from an HCI standpoint, with the translator as the central object of investigation (Castilho and O'Brien, 2016; Vieira, 2014; O'Brien, 2011). The CAT interface is thus the means by which data about the translator is collected, with the end goal being an enhanced understanding of the cognitive processes underlying translation. In some cases, HCI researchers also test the impact of new interactions or graphical components upon translators, by measuring productivity or cognitive load. A related line of investigation focuses upon scientific evaluation of the translation process from the standpoint of cognitive science (Carl, 2010; Koponen et al., 2012; Läubli et al., 2013; Läubli and Germann, 2015; Green et al., 2014b). This work typically conducts users studies with the goals of testing new interface components and/or modeling translator behavior.

On the other hand, MT and ML researchers working on interactive translation tools generally focus upon the data services which support translators' work, and the means by which they can be integrated into CAT workflows (see section 2.4 and chapter 5 for more detail). This involves testing proof-of-concept prototypes in user studies, or simulating user interaction with prototype components. For MT and ML researchers, CAT is a testbed for validating new predictive models and algorithm designs – if the model is good, this should be reflected in metrics correlated with translators' productivity. However, as already discussed, in the case of IMT and active learning, most existing research has used simulated user interaction to evaluate the effectiveness of the design (Ueffing and Ney, 2007; Ortiz-Martinez, 2011; Dara et al., 2014; Denkowski et al., 2014). In some cases, MT researchers have built real prototypes to test their ideas with human translators, but user studies are often not conducted, if the goal of the work is to introduce a new algorithm or training procedure for an MT system.

2.2.1 Interfaces for CAT Research

In the following sections, we review some of the user interfaces and prototypes that have been designed specifically for CAT research. We omit discussion of the major commercial CAT tools such as SDL-TRADOS, because we wish to focus upon research interfaces which have been used to test new functionalities for CAT in settings which are, at least theoretically, replicable and extensible by future researchers.

Transtype The Transtype project (Foster and Lapalme, 2000) conducted the first user-study on interactive machine translation, using an interface with the source sentence displayed in the top half of the window, and a composition area for the target translation in the bottom half of the window. The composition area provided a drop-down autocomplete containing an n-best list of suggestions for the next word in the translation.



Snapshot of a TT2 working session (English to Spanish translation). Text in normal typeface derives from TT2's predictions, while text manually typed by the translator is shown in light (red and underlined) font. In the last translation segment, the current best TT2 prediction appears inline in shaded text; the popup shows two alternative completions.

Figure 2.2: The Transtype 2 CAT interface (image from Casacuberta et al. (2009))

Transtype2 The Transtype 2 interface extended the interactive translation environment of Transtype to work with an actual SMT system (Barrachina et al., 2009). Figure 2.2 shows a screenshot of the Transtype 2 CAT interface.

MateCAT and CasmaCAT MateCAT (Federico et al., 2014) is a web-based CAT tool, created by a consortium of academic and commercial partners during a project funded by the European Union. MateCAT was developed for use in professional translation scenar-

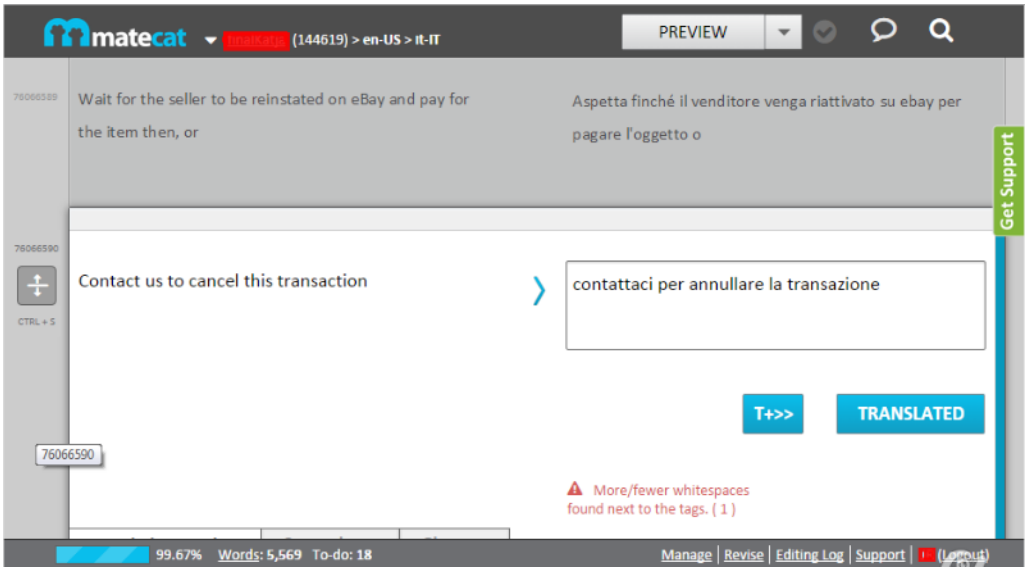


Figure 2.3: The MateCAT Interface, which is currently a commercial web-based CAT tool.

ios, while CASMACAT, a counterpart interface built upon the same underlying implementation, was developed specifically for CAT research (Alabau et al., 2014a). CASMACAT incorporates support for instrumentation, such as keystroke logging, which is important for experimentation. These tools are arguably the first mature open-source web-based CAT tools. CASMACAT in particular has been the platform of choice for research papers on diverse topics, including online learning (Alabau et al., 2014b), interactive machine translation (Alabau et al., 2013a; Underwood et al., 2014), and the integration of eye tracking into translation process analysis (Garcia Martinez et al., 2014). Figure 2.3 shows a screenshot of MateCAT.

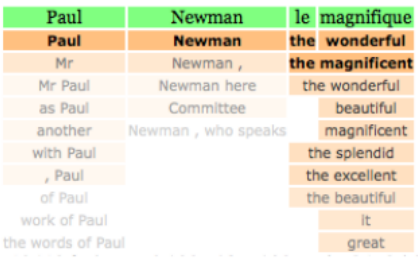


Figure 2.4: An example of the translation options rendered in CAITRA during interactive translation (Image from Koehn (2009)).

CAITRA CAITRA (Koehn, 2009c) was a unique interface for interactive translation. Translators see ranked suggestions below the translation window, where boxes correspond

to phrases, and rows correspond to the top paths through PB-SMT beam search. Translators compose the target translation by clicking on suggestions from the search lattice, or by typing in the target area. Figure 2.4 shows an example of the translation alternatives rendered by CAITRA.

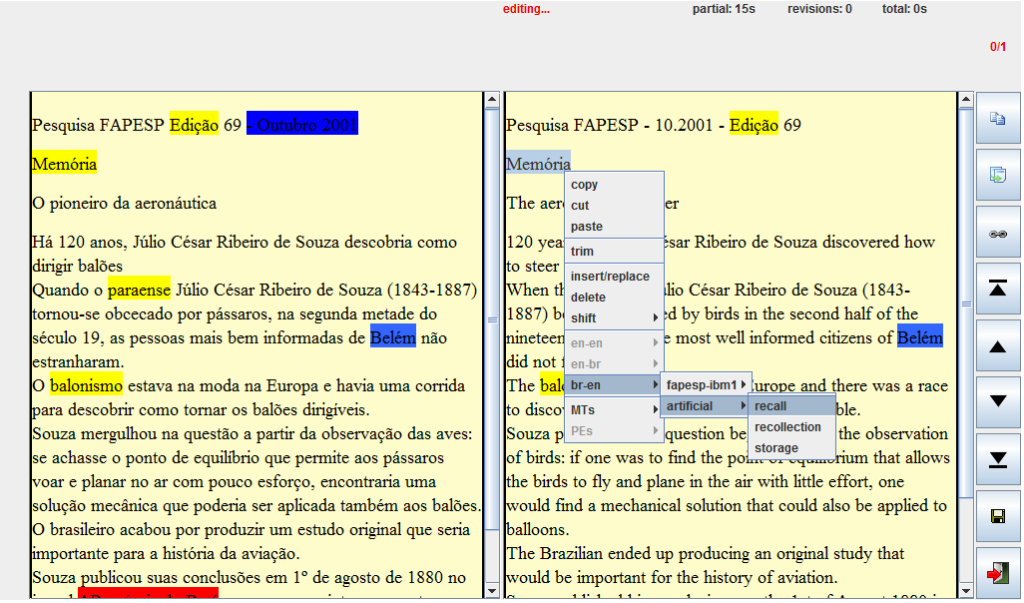


Figure 2.5: The PET interface.

PET The Post-Editing Tool (PET) (Aziz et al., 2012) is a simple interface for collecting statistics on the post-editing process. PET provides a toolset for annotating analyzing different error types found in translation, as well as computing metrics such as edit-distance. Figure 2.5 shows a screenshot of the PET tool.

OmegaT OmegaT is a popular open-source CAT tool, which has also been used in some user studies that implement new components and interaction modalities (Torregrosa et al., 2017). Figure 2.6 shows a screenshot of OmegaT.

2.2.2 Logging User Actions during Translation

Logging user behaviour is essential to CAT tool research. In additional to some logging functionality in several of the CAT research interfaces discussed above, several tools have been designed to help researchers log translator actions for analysis.

Translog Translog (Lykke Jakobsen, 1999) and Translog-II (Carl, 2012) are tools for recording and analyzing translators, created for the purpose of enabling sophisticated translation process research.

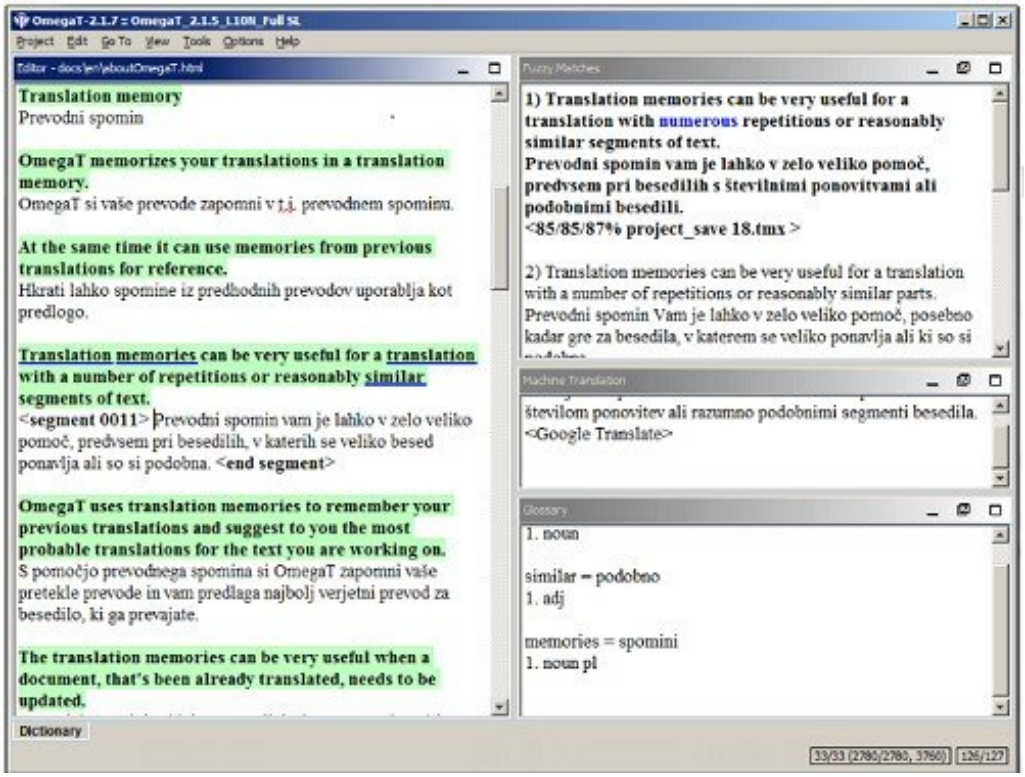

Project Edit Go To View Tools Options Help
Translation memory
Prevodni spomin
OmegaT memorizes your translations in a translation memory.
OmegaT si vaše prevode zapomni v t.i. prevodnem spominu.
At the same time it can use memories from previous translations for reference.
Hkrati lahko spomine iz predhodnih prevodov uporablja kot predlogo.
Translation memories can be very useful for a translation with a number of repetitions or reasonably similar segments of text.
<segment 0011> Prevodni spomin vam je lahko v zelo veliko pomoč, predvsem pri besedilih, v katerih se veliko besed ponavlja ali so si podobna. <end segment>
OmegaT uses translation memories to remember your previous translations and suggest to you the most probable translations for the text you are working on.
S pomočjo prevodnega spomina si OmegaT zapomni vaše pretekle prevode in vam predlaga najbolj verjetni prevod za besedilo, ki ga prevajate.
The translation memories can be very useful when a document, that's been already translated, needs to be updated.
1) Translation memories can be very useful for a translation with numerous repetitions or reasonably similar segments of text.
Prevodni spomin vam je lahko v zelo veliko pomoč, predvsem pri besedilih s številnimi ponovitvami ali podobnimi besedili.
<85/85/87% project_save 18.tmx >
2) Translation memories can be very useful for a translation with a number of repetitions or reasonably similar parts.
Prevodni spomin Vam je lahko v zelo veliko pomoč, posebno kadar gre za besedila, v katerem se veliko ponavlja ali ki so si
številom ponovitev ali razumno podobnimi segmenti besedila.
<Google Translate>
1. noun
similar = podobno
1. adj
memories = spomini
1. noun pl
Dictionary

Figure 2.6: The OmegaT interface.

iOmegaT and OmegaT-SessionLog iOmegaT (Moran et al., 2014) added instrumentation for logging translator actions into OmegaT, and was used to collect data from translators in an industry setting. OmegaT-Sessionlog[3] is an alternative logging plugin for OmegaT, which produces XML logs of translator actions.

2.3 Deep Learning for Machine Translation

Deep learning (DL) is a machine learning technique which allows continuous, distributed representations to be learned directly from data. This area of machine learning has its roots in work on modeling the activation patterns of actual neurons (McCulloch and Pitts, 1988) in the mid-1940s, and in models such as the perceptron (Minsky and Papert, 1969) and the multi-layer perceptron (Rumelhart et al., 1986a), but has recently seen a massive explosion in popularity across all areas of applied ML, and AI because of the flexibility of the model types that may be specified, their high performance on many tasks, and the exponential increase in available computational power, largely due to the use of Graphical Processing Units (GPUs), which can do large matrix multiplication operations orders of

[3] `https://github.com/mespla/OmegaT-SessionLog`

magnitude faster than CPUs.

Since the mid-1980s, *feed-forward* networks[4] have been a popular model choice for machine learning practitioners across many domains (Hornik et al., 1989), and are still used in many scenarios today. However, beginning in the late 1980s, *recurrent neural network* (RNN) architectures starting becoming a potential alternative architecture used to model natural language. RNNs are more flexible with respect to variable length sequences, and can outperform feed-forward models on some tasks (Elman, 1990; Bengio et al., 2003; Sutskever et al., 2014b; Schmidhuber, 2015). The long-short term memory (LSTM) cell (Hochreiter and Schmidhuber, 1997) and the more recent gated recurrent unit (GRU) cells (Cho et al., 2014b) have equipped RNNs with the ability to handle longer sequences, and to cope with long-distance dependencies in both inputs and outputs, important traits for models which deal with natural language. Finally, the invention of *attention mechanisms* (Bahdanau et al., 2014; Luong et al., 2015) has led to further improvements, especially for tasks which generate long sequences of text which are transformations of the input, such as MT. The menagerie of deep architectures that are currently used for different tasks is huge, and will only continue to expand as more researchers work on DL.

The word *deep* in DL refers to the fact that models are often composed of multiple layers of affine transformations followed by non-linear squashing functions, as opposed to shallow models, which consist of a single layer. However, in practice, the term deep learning is currently used to refer to most varieties of models based on neural networks, regardless of the number of layers a model actually has.

Schmidhuber (2015) gives a very thorough review of the history of DL with neural networks. Here we try to touch upon the key aspects of DL as it relates to the work in this thesis — we try to provide pointers to the most relevant research if the reader is interested in further exploring the very large body of work on this topic.

2.3.1 Sequence to Sequence Models

Sequence-to-Sequence (*seq2seq*) models (Sutskever et al., 2014c) have now become the de-facto approach of choice in many areas of NLP. These models are flexible, straightforward to implement, and scale well to very large datasets. The recent success of seq2seq

[4]We describe the terminology in this paragraph in detail below

can largely be attributed to the effectiveness of the computational structure of LSTM and GRU cells at capturing long-distance dependencies in sequences, and to the ability of attention mechanisms to maintain good output for large or complex model inputs. However, despite the good performance of these relatively new techniques, it is important to point out that researchers have been evaluating the use of RNNs for NLP, and specifically for machine translation for over 20 years (Chrisman, 1991; Waibel et al., 1991; Ñeco and Forcada, 1997; Castaño and Casacuberta, 1997), although with much smaller dataset sizes than are currently used.

We will now give an overview of key building blocks of seq2seq models as relates to the models used in this thesis. We then proceed to discuss the way these components may be assembled in current state-of-the-art models used for NMT.

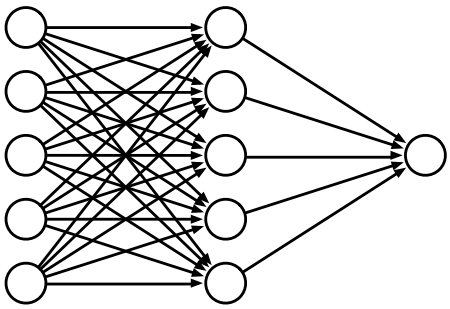


Figure 2.7: A fully connected feed-forward network with two layers and a single output value (image from Koehn (2017)).

Feed-forward Layers

The feedforward layer can be described as a linear transformation of a vector $\mathbf{x}$, plus a bias[5], when the $\mathbf{b}$, i.e. $f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ followed by an element-wise non-linearity $g()$ where $g()$ is usually one of Eqs. 2.1a–2.1c:

$$\text{logistic}(x) = \frac{1}{1 + e^{-x}} \tag{2.1a}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.1b}$$

[5]Note that the purpose of the bias is to shift the linear function implemented by the parameters of the feedforward layer. Some descriptions of neural networks omit the bias term if input data is normalized, or a layer uses a "global" normalization such as the softmax function (see below).

$$\text{ReLU}(x) = \mathbf{max}(0, x) \tag{2.1c}$$

Note that we are using x to denote a single element of the vector $\mathbf{x}$, since each of these functions is an element-wise operation.

A single feed-forward layer can thus be described by Eq. 2.2:

$$\text{feedforward}(\mathbf{x}) = g(f(\mathbf{x})) = ReLU(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.2}$$

Feedforward layers may be stacked to create deeper models:

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{2.3a}$$

$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{h_1} + \mathbf{b}_2) \tag{2.3b}$$

$$\ldots \tag{2.3c}$$

where the parameters $\mathbf{W}_i$ and $\mathbf{b}_i$ are different for each layer. Stacking often leads to better model performance, and the use of > 1 layers in a feedforward network yields the (theoretical) capacity to approximate any function (Hornik et al., 1989).

Embedding Layers

The embedding layer is the interface between symbolic units used in language, and the dense, real-valued vectors used by neural networks. The parameter matrix $\mathbf{W}_{\text{emb}}$ of an embedding layer stores a real-valued vector for each possible input symbol. One of the key innovations of of neural networks is that the dense feature representations for the input symbols are *parameters* of the model, and are thus learned directly from data. Before the practice of embedding became commonplace, the first step in a machine learning pipeline was necessarily the manual or heuristic extraction of representative features of the input. However, this way of extracting features requires researchers with domain expertise to pre-specify which attributes of the input are important, which is difficult, time-consuming, and error-prone. Word2Vec (Mikolov et al., 2013), popularized the approach of training

input features automatically, and embedded representations of input tokens are now a standard component of almost every state-of-the-art NLP model for any task.

For a sequence of symbols $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$, we specify that each $\mathbf{x}_i$ should be a "one-hot" vector of size $|V|$ – there is a 1 at the index of the symbol, and zeros everywhere else. We may then multiply this vector with the embedding matrix $\mathbf{W}_{\text{emb}}$ to obtain the embedding of input symbol x_i (Eq. 2.4):

$$\text{embedding}_i = \mathbf{W}_{\text{emb}}\mathbf{x}_i^T. \tag{2.4}$$

In other words, the embedding of symbol x_i corresponds to the column of $\mathbf{W}_{\text{emb}}$, which is indexed by the single "1" value in x_i, a unique index representing this symbol. For example, x_i might be the word "dog", and the index of "dog" in a lookup dictionary might be 75. embedding_i would then correspond to the 75$^{\text{th}}$ column of $\mathbf{W}_{\text{emb}}$. The size of the embedding ($|\text{embedding}|$) for each symbol is a hyperparameter, and $\mathbf{W}_{\text{emb}}$ is $\mathbb{R}^{|\text{embedding}|\times|V|}$.

2.3.2 Recurrent Layers

Recurrent layers, as the name implies, introduce a time-recurrent computation structure. Elman (1990) introduced a simple architecture with a recurrent property using three sets of parameters, $\mathbf{W}_h$, $\mathbf{U}_h$, and $\mathbf{W}_y$ which correspond to *hidden*, *recurrent*, and *output* matrices respectively (these matrices also have bias vectors, shown below). The computational structure of an Elman network, or SRN, is shown in Eq. 2.5:

$$\mathbf{h}_t = \sigma_{\mathbf{h}}(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.5a}$$

$$\mathbf{y}_t = \sigma_{\mathbf{y}}(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_y) \tag{2.5b}$$

where $\mathbf{h}_t$ is the hidden state at the current timestep, $\mathbf{y}_t$ is the output at the current timestep, and σ denotes one of the *activation functions* (Eqs. 2.1a–2.1c).

Recurrent models maintain a running representation of the history thus far, using $\mathbf{h}_t$ to keep track of model state at timestep t, while re-using the same set of parameters to

combine the current model state with new inputs. In the Elman SRN formulation, there is also an *output* transformation which yields $\mathbf{y}_t$, this can be seen as the "prediction" of the model at timestep t. Depending upon the application where RNNs are being used, we may only use the final $\mathbf{y}_T$ as output, or we may use the output of every $\mathbf{y}_1 \dots \mathbf{y}_T$. Other types of recurrent networks, such as LSTMs and GRUs, maintain the fundamental parameter sharing and stateful notions of SRNs, while introducing gates which control the respective contributions of the input and the current state to the computation of the model state at the next timestep. See, for example Hochreiter and Schmidhuber (1997),Cho et al. (2014a), or Goodfellow et al. (2016) for more detail on these RNN extensions.

2.3.3 Training Deep Neural Networks with Backpropagation

Before proceeding to discuss how these components can be assembled into a model for NMT, we need to mention how the parameters θ of a model are learned from training data. The mapping of the outputs of a DL model into a probability distribution using the logistic function or, in the case where the model has more than one output, the $\mathbf{softmax}$ function, is discussed in detail in chapters 5 and 6, so we omit these equations here.

Now we will show how to learn the parameters of model which outputs sequences $\mathbf{y} = \mathbf{y}_1 \dots \mathbf{y}_N$, corresponding to the second RNN use case which makes an output prediction at every timestep, discussed in the previous section. If we view the output of our model as a probability distribution over the possible outputs $\mathbf{y}$, i.e. $P(\mathbf{y}|\mathbf{x};\theta)$, which we want to learn using a training dataset $\mathbf{D} = \Big\{\langle \mathbf{x}^{(s)}, \mathbf{y}(s)\rangle\Big\}_{s=1}^{S}$, then optimization can be seen as the search for the parameters $\hat{\theta}$ which maximize the probability of the training data (Eq. 2.6):

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \Big\{\mathcal{L}(\theta)\Big\}, \tag{2.6}$$

where $\mathcal{L}(\theta)$ is some differentiable function of $\mathbf{x}$, $\mathbf{y}$, and θ. The standard instantiation of $\mathcal{L}(\theta)$ for sequence prediction tasks looks to maximize the log-likelihood of the training

data, so $\mathcal{L}(\theta)$ expands to Eq. 2.7:

$$\mathcal{L}(\theta) = \sum_{s=1}^{S} \log P(\mathbf{y}|\mathbf{x};\theta) \tag{2.7a}$$

$$= \sum_{s=1}^{S} \sum_{n=1}^{N^{(S)}} \log P(\mathbf{y}_n^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}_{<n}^{(s)};\theta). \tag{2.7b}$$

where $\mathbf{N}^{(s)}$ is the length of target sentence $\mathbf{y}^{(s)}$. Equation 2.7 says that the total loss for our model is the log loss computed for every token $\mathbf{y}_n^{(s)}$ in every target sentence $\mathbf{y}^{(s)}$, and the second part of Eq. 2.7 is obtained by factorizing the probability of each item in $\mathbf{y}^{(s)}$ to depend only on $\mathbf{x}^{(s)}$, and the preceding items $\mathbf{y}_{<n}^{(s)}$. This factorization is discussed at length in chapters 5 and 6.

We can now look for $\hat{\theta}$ from Eq. 2.6 using an optimization method of our choice. Gradient descent (GD), which optimizes parameters based on the entire dataset $\mathbf{D}$ and stochastic gradient descent (SGD) which only optimizes based upon samples drawn from $\mathbf{D}$, are the most popular methods of optimization because they are easy to implement and tend to work well in practice. Here, for clarity, we have defined a function that we wish to *maximize*, so we will do gradient *ascent*, but the two techniques are functionally equivalent, and differ only by a negative sign.

In order to use gradient-based methods to optimize the parameters of our model, we need to know how much each parameter contributes to the current value of $\mathcal{L}(\theta)$. The partial derivative of a single parameter θ_i is Eq. 2.8:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i} = \sum_{s=1}^{S} \sum_{n=1}^{N^{(S)}} \frac{\partial P(\mathbf{y}_n^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}_{<n}^{(s)};\theta)/\partial \theta_i}{P(\mathbf{y}_n^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}_{<n}^{(s)};\theta)}, \tag{2.8}$$

which we can use to update our parameter at each iteration by taking a small step in the direction that will make $\mathcal{L}(\theta)$ bigger (Eq. 2.9):

$$\theta_i^{t+1} = \theta_{\mathbf{i}}{}^{t} + \alpha \frac{\partial \mathcal{L}(\theta)}{\partial \theta_i}, \tag{2.9}$$

where α indicates the *learning rate*, which is a training hyperparameter.[6]

In deep feedforward networks, and in recurrent models, the computation of later layers depends upon the output of previously computed layers, which ultimately depend upon the input **x**. The error of the current model is computed using the representation obtained by transforming **x** using the model's parameters, and the true output **y**, which we know for each training instance. For models with millions of parameters, independently computing equation 2.8 for each θ_i would be intractable and extremely slow. The *backpropagation algorithm* (Rumelhart et al., 1986b; Chauvin and Rumelhart, 1995; Schmidhuber, 2015) is an exponentially more efficient way to use a single forward pass through the model, along with the errors computed starting at the last layer and working backward toward the input, to obtain $\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i}$ for every parameter. In practice, almost all deep learning models use the backpropagation algorithm to obtain gradients during training (Schmidhuber, 2015).

By viewing the computations of a model as a directed graph, the dependency patterns between different parameters can be made explicit. Modern deep learning frameworks such as Theano[7], Tensorflow[8], and PyTorch[9] structure models as directed computation graphs, which allow these frameworks to employ *automatic differentiation* (Baydin et al., 2015). Using this technique, model developers do not need to specify the way that gradients are computed, as long as the model can be expressed as a directed computation graph. This drastically improves development speed, and reduces the chances for errors or software bugs – thus, libraries with such functionality are used by most researchers working on applications of DL.

In the following section, we give a brief introduction to the most popular architecture for NMT, which is the basis for many of the models discussed in this thesis.

2.3.4 Neural Machine Translation

Over the last several years, NMT has become the most active sub-topic of MT research. The performance improvements given by NMT, as well as the flexibility and extensibility of deep learning models in general have led to a very quick adoption, with NMT replacing legacy SMT systems in both academic and industry scenarios (Wu et al., 2016; Junczys-

[6] We have mostly followed the notation of Shen et al. 2016.
[7] http://deeplearning.net/software/theano/
[8] https://www.tensorflow.org/
[9] https://pytorch.org

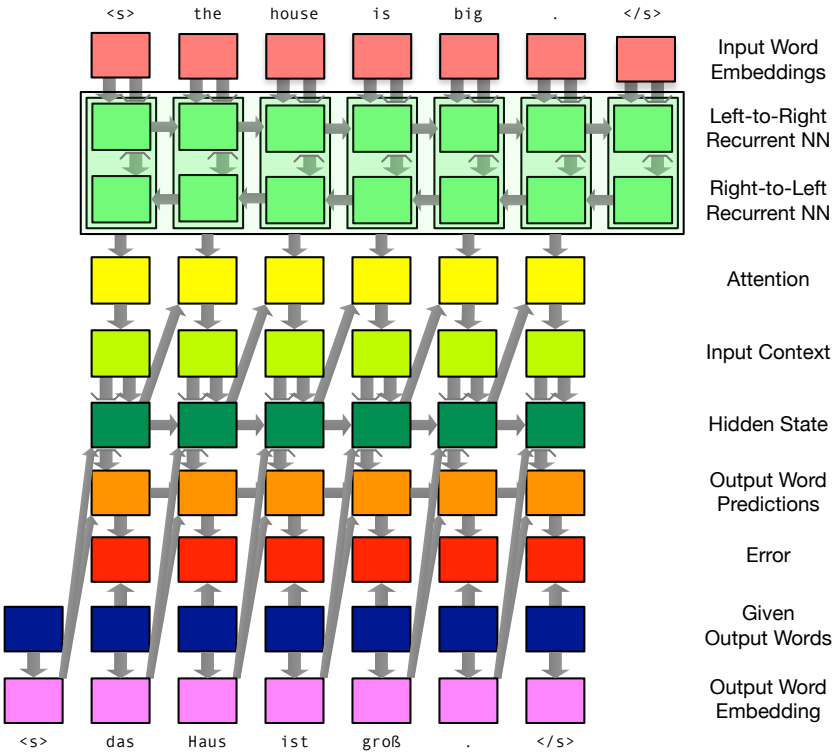


Figure 2.8: A schematic of an encoder decoder model with attention for NMT (image from Koehn (2017)).

Dowmunt et al., 2016; Koehn, 2017).

In the context of sequence to sequence models, the MT task can be viewed as a process of first *encoding* the source segment into an dense latent representation, then *decoding* the target sentence from this space using the representation of the source (Sutskever et al., 2014a; Cho et al., 2014b).

The fundamental challenge in applying deep learning to machine translation is that the source and target sequences are of different lengths; thus, network architectures must be designed which can embed variable-length sequences into fixed-length representations. Recent applications of DL to MT implement this idea in different ways, see for example (Sundermeyer et al., 2014; Devlin et al., 2014; Bahdanau et al., 2014; Kalchbrenner and Blunsom, 2013). The primary difference between these methods is the means of encoding the source sentence: Kalchbrenner and Blunsom (2013) uses an adaptation of the convolutional model which is common for image processing, while Cho et al. (2014a) and Bahdanau et al. (2014) use an RNN. All of these methods use an RNN to generate the target sequence from the source representation.

Figure 2.8 depicts an encoder-decoder model with attention (Bahdanau et al., 2014) –

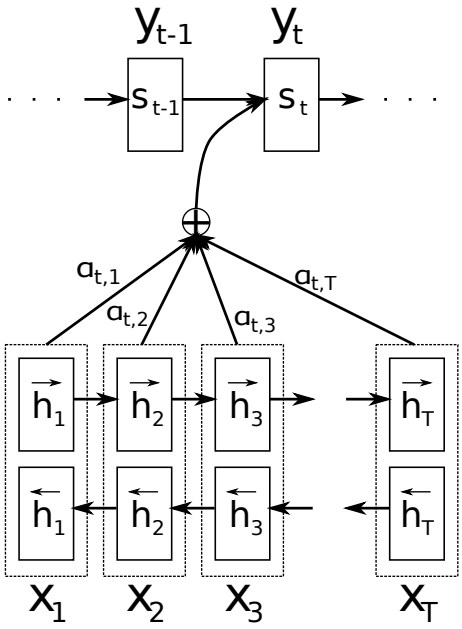


Figure 2.9: The "RNN-Search" (Encoder-Decoder with Attention) architecture (image from Bahdanau et al. (2014)).

the input is shown at the top of the figure, and the output at the bottom. In the preceding sections, we have introduced every component of this model, except *attention*, which is discussed in the next section.

2.3.5 Attention

Encoding the entire source sequence $\mathbf{x}$ into a fixed representation may introduce irrelevant information in some cases, because many translation decisions only depend upon one or a few words in the source segment (Devlin et al., 2014; Bahdanau et al., 2014) Figure 2.9 depicts a model where the decoder RNN conditions its output upon a dynamically computed weighted sum of the encoder RNN hidden states, where the attention weights α are recomputed at each decoding timestep using the internal state of the decoder.

Recently, *attention* models (Bahdanau et al., 2014; Luong et al., 2015) have been shown to perform better than fixed representations, because they allow a model to learn to dynamically focus only on the most relevant information in the input at each step of generating the output. The intuition behind this idea is that, at each generation timestep, an MT model should be able to select only the information in the source sequence which is relevant to generation of the current token, and ignore the rest. Attention mechanisms are now a de-facto component of all current state-of-the-art NMT models, regardless of the specific architecture (Koehn, 2017). The equations of the attention mechanism are

discussed in detail in chapter 5, so we omit them here.

2.4 Interactive Text Prediction

The vision of translation environments as collaborative interfaces which connect humans and machines, taking advantage of their respective strengths to efficiently produce high-quality translations, is almost as old as the field of MT itself (Kay, 1997; Green et al., 2015). As has already been discussed, interfaces that facilitate such a collaboration between translators and MT systems are often called IMT systems. However, two distinct lines of research have recently adopted the IMT moniker — one using it to mean user-guided machine translation, especially *interactive-* or *guided-* decoding (Ueffing and Ney, 2005; Barrachina et al., 2009; Casacuberta et al., 2009; Koehn, 2009b; Ortiz-Martinez, 2011; Green, 2014), which Foster et al. (1997) refers to as "target-text mediated" interaction, the other using it to mean online or batch retraining of SMT systems using feedback from human translators (Haffari and Sarkar, 2009; Denkowski et al., 2014; Martínez-Gómez et al., 2012; Mathur et al., 2013). Although there is some connection between these two research directions, this conflation of distinct topics is confusing; thus, *interactive text prediction* (Foster, 2002; Green, 2014) or *interactive-predictive machine translation* (Ortiz-Martinez, 2011; Casacuberta et al., 2014) are less ambiguous ways to describe the former case, which is one of the broad topics of this thesis[10].

The key aspect of IMT that differentiates it from other uses of MT in translation is that it is *interactive* (figure 2.10). IMT can be viewed as a collaboration between an expert (the translator), and an industrious but somewhat overzealous assistant (the MT system). The expert's job is to guide the MT system towards good translations by providing feedback at key points in the translation process, while the MT system's job is to be as helpful as possible without getting in the way. As discussed in the previous section, helpful in this context means that the system saves the translator some work, either by auto-completing an already-planned translation, or by providing high-quality suggestions, thus reducing the cognitive load on the translator, and possibly shortening the overall time-to-completion. Green (2014) emphasizes the distinction between the high-precision actions

[10]Throughout this thesis, the abbreviation *IMT* should always be understood in the sense of interactive text prediction unless explicitly indicated otherwise.

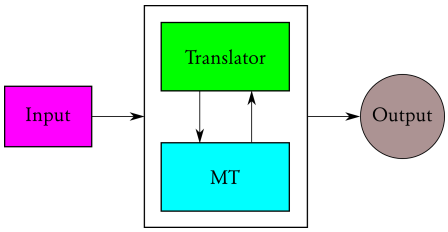


Figure 2.10: Interactive translation

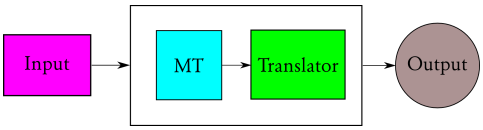


Figure 2.11: Post-editing

of humans, which lead to very good translations but may take a long time to complete, and the high-recall of machines, which can provide outputs of unknown quality at very high speed.

Interactive translation vs. post-editing In contrast to interactive translation, post-editing tasks, such as the revision of MT system outputs or TM fuzzy matches, force translators to view the *translation provider* as a black box, and to cope with whatever this upstream system outputs (figure 2.11). The translator has no control over the system, and must act as a sort of gatekeeper, ensuring that low quality MT output is transformed to human quality output, but performing no modification of MT output that is already at a passable standard. Another important contrast between interactive translation and post-editing lies in the way the translator provides feedback on translation quality to the interface: in IMT, each part of each translation is *explicitly* confirmed; in post-editing, by doing nothing, the translator tacitly indicates that MT output which is left unchanged is "good enough", while any errors in translation are implicitly annotated as such, because the translator must fix them[11].

The human-in-the-loop design of IMT means research must necessarily be conducted on two fronts:

1. Designing algorithms, models, and frameworks that allow users to intervene in and guide the MT decoding

2. Designing the user interface components that allow this interaction to take place

In this thesis, we study both of these angles: chapters 3 and 7 consider user interfaces designed for interactive translation, while chapters 5 and 6 present models and algorithms

[11]This implicit annotation of errors from Post-Editing tasks is put to use in the creation of training datasets for word-level QE, and APE, which are discussed in the following section of this chapter, and in chapter 4.

for MT models that can be deployed in interactive scenarios.

The MT-Translator Interface

Foster et al. (1997) introduced the concept of "target-text mediated IMT". Prior to this work, interactive translation had been proposed via source text modification (Zajac, 1988), also known as "pre-editing" (Hutchins, 1998). Pre-editing was especially attractive for rule-based MT (RBMT) systems — by transforming source inputs to fit within a fixed set of templates, a system operator could be sure that an MT system would be able to translate each input. An additional advantage of this approach was that it could theoretically be done by monolingual operators (Hutchins, 1998). However, the tedium and brittleness of designing and maintaining RBMT systems, and the success of SMT beginning in the early 90s (Brown et al., 1988, 1993), led to new possibilities for interactive designs. Interaction via the target text has now become the de-facto method of implementing IMT. Translators interact with a machine translation system by modifying target text; certain actions, such as typing a space character, signal that the MT system should provide an extension of the current prefix.

2.4.1 IMT Mathematical Background

We use the terms IMT and *text prediction* interchangeably to denote the task of predicting the continuation of a target-language string conditioned on some contextual information. In other words, in IMT we are trying to find the best *suffix* given a *prefix*, and possibly some additional *context*. In the simplest case this corresponds to finding the prediction $\hat{c}$ which satisfies equation. 2.10:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c|\mathrm{prefix}) \tag{2.10}$$

Where C is the set of all possible completions for prefix.

The basic instantiation of this task is the monolingual scenario, where a standard language model can be used to generate predictions given a prefix. However, note that, even for the translation use cases we consider below, this simple monolingual auto-completer, which has no information about the source and merely conditions its predictions on the target context, may save the user some keystrokes, especially as the conditioning prefix

grows longer, providing more information about the target translation without reference to the source segment. It is thus useful to consider the monolingual model as the baseline, and to try to improve incrementally by progressively giving this model access to more contextual information.

In prefix-based IMT, the prefix is assumed to be a partially-completed target-language translation of a complete source-language input. In addition to the prefix itself, the prediction of the continuation can also be conditioned on other contextual information (Foster, 2002; González-Rubio et al., 2013; Green et al., 2014a). The finished translation will fully cover the source input, so all of the content in the source will have a corresponding translation in the final output. This notion of *coverage* is central to the search phase of MT, and will resurface at several points in our discussion.

Our predictions should clearly also be conditioned upon the source sentence, because this information is likely to make the model's guesses about how the translation should continue much more accurate. We modify equation 2.10 to also include the source sequence (equation 2.11). Therefore, in the most general sense, a text prediction model is a special kind of language model, which conditions on both the preceding context of the target sequence, and the source input.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|\mathrm{prefix}, \mathrm{source}) \tag{2.11}$$

In the formulation typically used for IMT, the search for the completion c that maximizes equation 2.11 can alternatively be viewed as a constrained version of classic MT, which searches for the best translation given the source, with the additional constraint that all outputs must contain the prefix.

With this high level view of the search problem which IMT systems try to solve, the following sections give an overview of important avenues of interactive translation research, summarizing existing approaches. The next sections, as well as Chapters 5 and 6 go into more depth and extend this basic framework in various directions.

Text Prediction with SMT

Research on target-side text prediction has evolved from maximum entropy classification models (Och and Ney, 2002; Foster, 2002), which select predictions from a static vocabulary, into interactive decoding, where an SMT decoder is modified to conduct a constrained search over the possible suffixes of a user-provided prefix.

Foster (2002) was the first to present a complete framework for target text prediction within translation interfaces. He proposed the maximum entropy/minimum divergence (MEMD) translation model, which can take into account features from both the source text and the existing target prefix while also predicting the length of the completion, and the probability that the user will accept a completion. However, Foster's maximum entropy framework can only chose the best single continuation phrase from a lexicon, because it does not include a decoding component which can compose output sequences from smaller units.

Ueffing and Ney (2005), Koehn (2009b), Barrachina et al. (2009), Ortiz-Martinez (2011), and Green (2014), all focus upon tight integration between the translation interface and the SMT system. The IMT models in their work are purpose-built SMT systems which perform prefix decoding by taking advantage of the coverage-based beam search[12] used in phrase-based MT (Koehn, 2010b). This method of predicting the most probable suffix given an input prefix is limited to models which use coverage-based beam search, restricting their applicability to a certain class of SMT models.

Using the notation of the standard log-linear framework for phrase-based SMT (PB-SMT) (Och and Ney, 2002; Koehn, 2010b; Green, 2014), the task of interactive prediction system is to search for:

$$\hat{e} = \underset{e \in \text{prefix}(e,h)}{\text{argmax}} \ \mathbf{w}^T \Phi(e, f) \tag{2.12}$$

Where $\text{prefix}(e, h)$ indicates that h is a prefix of the target hypothesis e, f is the source sequence, ϕ is a function which returns a vector containing the values of each of the feature functions in the log-linear model, and w is a vector containing scalar weights for each of these features. Note that this is quite similar to Eq. 2.11, but returns an unbounded

[12] see chapter 6 for a discussion of different beam search types

score, instead of a probability. The expanded form of the $\mathbf{w}^T\Phi(e, f)$ term is:

$$\mathbf{w}^T\Phi(e, f) = \sum_{i=1}^{N} w_i\phi_i(e, f) \tag{2.13}$$

Where N is the number of feature functions in the model. Importantly, the optimal values for the $\mathbf{w}$ parameters must be learned on held-out data, in the training phase called *tuning*.

Because our work is focused upon NMT, we provide a short introduction to prefix-decoding with NMT below. In chapters 5 and 6 we provide a more detailed discussion of prefix decoding for IMT, and we propose generalizations of IMT beyond prefix decoding in chapter 6.

Handling Unknown Tokens in IMT Prefixes

Ueffing and Ney (2005), Barrachina et al. (2009), Ortiz-Martinez (2011), Koehn (2009b) rely on the word-graph or *lattice* (Koehn, 2010b) produced by the SMT system after decoding the source-language segment. The lattice contains all possible translation paths from beginning to end, with weighted edges indicating the model score. To use such a lattice for IMT, we must first locate the user-provided prefix, then find the best-scoring path starting from the end of the prefix. However, approaches based upon completed word graphs all suffer from an important shortcoming: they cannot handle arbitrary out-of-vocabulary (OOV) prefixes which were not observed during training. In real-world translation scenarios the input tokens are likely to come from an unconstrained vocabulary, and may contain irregularities such as spelling errors, or be domain-specific vocabulary that was not observed in the training data. Thus, in practice, most IMT systems need to be equipped with a robust way to able to handle unseen tokens.

Some SMT-based IMT systems have attempted to overcome this problem using fuzzy-matching via string-edit distance in cases where the target prefix cannot be exactly matched (Barrachina et al., 2009; Ortiz-Martinez, 2011). Green et al. (2014a) presents a more satisfying approach, which dynamically adds artificial phrases to the SMT phrase table. When an unseen token is encountered, all possible source alignments are added to the SMT phrase table with low, constant translation feature scores, allowing the system to

propose completions which are based on OOV target prefixes. Some of the NMT models discussed in the next section can also overcome this shortcoming by using sub-word units or character-level representations.

Text Prediction with NMT

As discussed in section 2.3, the most popular method for training NMT models is to maximize the log-likelihood of the training corpus (equation 2.14).

$$\theta^\star = \arg\max_\theta \sum_{(X^i,Y^i)} \log P(Y^i|X^i;\theta) \ . \tag{2.14}$$

This optimization is accomplished with stochastic gradient descent over mini-batches of (X^i, Y^i) pairs, where each $Y^i = y_1, y_2, \ldots, y_T$ is a reference translation of the corresponding $X^i = x_1, x_2, \ldots, x_T$. In the following discussion we omit the training corpus index superscripts for simplicity ($Y = Y^i$ and $X = X^i$) unless explicitly indicated. The (log) probability of an output Y is modeled as equation 2.15:[13]

$$\begin{aligned} \log P(Y|X) &= \log P(y_1^T|X) \\ &= \sum_{t=1}^{T} \log P(y_t|y_1^{t-1}, X) \end{aligned} \tag{2.15}$$

Thus, the probability of output y_t at timestep t is conditioned on both X, and the current prefix y_1^{t-1}. After training our model to obtain $\theta^\star$, at decoding time, we can ask the model to predict the best output symbol at the current timestep ($\hat{y}_t$) with equation 2.16:

$$\hat{y}_t = \underset{y_i \in \{\mathbf{v}\}}{\operatorname{argmax}} P_{\theta^\star}(y_i|X; y_1^{t-1})), \tag{2.16}$$

As equations 2.15 and 2.16 show, both the NMT training and decoding objectives factor $P(Y|X)$ into local decisions about the current y_t, given the input, and the current prefix. Each y_t depends upon the y_1^{t-1} symbols preceding it, but not upon any $y_t \in y_{t+1}^T$. This is an extremely convenient property, because it means that NMT models can be directly

[13] We use the notation from Bengio et al. (2015)

used for IMT, with no modification of the training procedure whatsoever. In practice, re-purposing an NMT model for IMT only requires a small change to the decoding logic, allowing the decoding model to first run through a user-provided y_1^{t-1}, instead of generating every $y_1 \dots y_T$ using $P_{\theta^\star}$. NMT is thus well-suited to prefix-constrained text prediction, because the decoding objective at each timestep exactly matches the decoding objective of equation 2.11.

Recent work has capitalized upon this similarity, performing the simple adaptation described above to re-purpose NMT frameworks to also support prefix-decoding. We also use this idea and extensions thereof in the experiments described in chapter 5.

Note that the standard MT decoder can be abstractly written as $d(X) = Y$, in other words, a function $d()$ that takes a source sequence X, and returns a target sequence Y. Both Knowles and Koehn (2016) and Wuebker et al. (2016) change the decoding interface to pre-trained NMT models to be $d(X, y_1^{t-1})$. As described above, this only requires a modification to the decoding logic — no retraining or modification of the objective function are needed. They then evaluate the performance of pre-trained NMT models in an IMT context, an approach which we also employ in chapter 5.

2.4.2 Interactive Post-Editing

Almost all work on interactive translation has exclusively considered *prefix-based* IMT. However, some recent work has proposed more flexible scenarios, where users can interact with any span in the output.

Recently, there has been interest in extending interactive translation beyond the prefix-based designs discussed above. Broadly, we wish to allow translators to modify any portion of a translation hypothesis, while holding other parts fixed. Without interactivity, this idea is exactly the post-editing task discussed in previous sections. Although post-editing has been shown to be faster than both IMT and translation from scratch, an important problem with post-editing is that many translation errors are not localized to a specific part of the hypothesis, because words and phrases in a translation are not independent of one another. Fixing an important error in a translation could make other required fixes trivial to solve, or could point to the need for a complete syntactic restructuring of the translation. Thus we would like translators to be able to make targeted *local* edits that an

MT system can then use as guides in the search for the globally best translation. We call this scenario *interactive post-editing* (IPE): a translator edits part of a hypothesis, then asks the model for a new output which maintains these edits, while revising the rest of the translation to take the edits into account. The localized edits made by translators can thus be regarded as *constraints* on the final output. Chapter 6 presents our contributions to this novel research area — the following section gives an overview of important related work.

The IMT designs discussed above can be viewed as sub-case of constrained decoding, where there is only one constraint which is guaranteed to be placed at the beginning of the output sequence. Recently, some attention has also been given to SMT decoding with multiple lexical constraints beyond prefixes. The pick-revise machine translation (PRIMT) (Cheng et al., 2016) framework for IPE introduces the concept of *edit cycles*. Translators specify constraints by editing a part of the MT output that is incorrect, and then asking the system for a new hypothesis, which must contain the user-provided correction. This process is repeated, maintaining constraints from previous iterations and adding new ones as needed. Importantly, their approach relies upon the phrase segmentation provided by the SMT system (indirectly an artefact of the word-alignment model used by the system). The decoding algorithm can only make use of constraints that match phrase boundaries, because constraints are implemented as rules which enforce that source phrases must be translated as the aligned target phrases that have been selected as constraints. In contrast, our approach, presented in chapter 6, decodes at the token level, and is not dependent upon any explicit structure in the underlying model.

Domingo et al. (2016) also consider an interactive scenario where users first choose portions of an MT hypothesis to keep, then query for an updated translation which preserves these portions. The MT system decodes the source phrases which are not aligned to the user-selected phrases until the source sentence is fully covered. This approach is similar to the system of Cheng et al., and uses the XML input feature in Moses to enable the functionality (Koehn et al., 2007).

Some recent work also considers the inclusion of soft lexical constraints directly into deep models for dialog generation, and special cases, such as recipe generation from a list of ingredients (Wen et al., 2015; Kiddon et al., 2016).

In chapter 6 we present a flexible framework for including arbitrary lexical constraints into NMT decoding. Our solution allows different constraints to be reordered with respect to one another, but guarantees that every constraint will be present in the model's output.

Generating Suggestions in Real-time During Interactive Translation

When using IMT systems in realistic scenarios, the runtime performance of the system must be taken into account. The basic measurement of IMT runtime performance is the time it takes to predict one word of a suffix. Some portion of this time will be baked in to the system, i.e. the time it takes to do a matrix multiply. However, there are two primary runtime parameters that can be tweaked to make performance better, usually with a tradeoff in output quality.

- decoder beam size
- maximum length of predicted suffix

By reducing both of these, prediction time can be decreased until it meets a specified threshold (Wuebker et al., 2016; Knowles and Koehn, 2016). We discuss decoding-time complexity again in chapters 6 and 7.

2.4.3 User Interfaces for Interactive Translation

As we have already mentioned, the intuition motivating the design of interactive translation systems is that the user can be asked to perform the tasks which are the most difficult for an automatic translation system, while the system provides lookup and autocompletion functions that reduce the cognitive load for the user. In general, users are asked to perform precision-oriented tasks, while recall-oriented tasks are automated (Green et al., 2014b).

When a user is interacting with an MT system during the decoding process, one or more decisions, such as choosing the best continuation given an existing prefix, are performed by the translator. This sequential exchange of control between translator and MT system is captured by the *mixed-initiative* interactive paradigm discussed in section 2.1.1.

Early work showed a substantial decrease in productivity in comparison to traditional post-editing (Langlais et al., 2000). More recent prototypes have shown that IMT can be

significantly more efficient than manual translation[14], especially when the embedded MT system generates high-quality completions (Green et al., 2014a; Green, 2014; Barrachina et al., 2009; Koehn, 2009b; Casacuberta et al., 2009). Green et al. (2014b) directly compared translator productivity in PE versus IMT scenarios, and found that IMT is not faster than post-editing, but that the quality of translations produced with IMT can be better than that of post-edited translations.

Even if interactive translation cannot beat post-editing in measures of sheer productivity, there are good reasons for continuing text prediction research in the context of CAT. The left-to-right[15] interaction paradigm mimics the manual composition process much more closely than the post-editing task, which drastically alters the cognitive model of the translation task (Carl, 2010). Interactive translation is also especially appealing for lower-resource languages and domains, where the MT system output may not be of sufficient quality to enable efficient post-editing, but may be able to continually provide short extensions to an existing target prefix, saving the translator some effort in completing the translation.

Allowing direct interaction between humans and translation systems intuitively simplifies the MT task in some ways, because MT system can delegate when prediction confidence is low, asking a human to make the most challenging and risky decisions, potentially limiting the opportunities for MT to make severe errors. We test this hypothesis extensively in chapter 5, but our results do not point to low-hanging fruit for such risk-aware IMT.

Designing user-facing components for IMT is also very challenging, especially because the system must be able to respond to user queries very quickly to be any use in realistic translation scenarios. For this reason, all prototypes to date have made major simplifying assumptions to the theoretical models of interaction in order to make the problem tractable for the available computational resources, such as limiting the stack size of the decoder, and using small phrase tables extracted for specific test datasets.

In order to perform interactive translation, we must provide the user with the following functionality at at minimum:

[14] As already discussed, most of this work uses simulated editing, assuming a flawless manual translation, where the user types each character in the reference exactly

[15] or right-to-left, depending upon the target language

1. viewing the output of the system

2. interacting with system output (i.e. accepting or rejecting suggestions)

3. asking the system for new output

Several graphical interface components have been proposed to meet these requirements. The most common is a drop-down element which moves with the user's cursor (Foster, 2002; Green et al., 2014a; Torregrosa et al., 2017). Green also includes several design enhancements which improve the interface between translators and the SMT system, including limiting the number of suggestions, and providing keyboard shortcuts for opening and closing the dropdown. Barrachina et al. (2009) uses the target text directly — when the user places the cursor at a position in the target text, a new suggestion is generated using the text to the left of the cursor as a prefix. Koehn (2009b) provides a novel interface which displays a pruned lattice to the translator, who completes the translation by selecting phrases from left-to-right. Sanchis-Trilles et al. (2008) propose using mouse actions as the signal that the SMT system should provide a new suggestion which extends the prefix to the left of the cursor position.

2.4.4 Truncating Hypotheses to Maximize Expected Benefit

In text prediction environments which generate a suffix given a prefix $y_1 \dots y_{t-1}$, uncertainty grows as the length of the prediction increases. However, current IMT models do not explicitly take this uncertainty into account. A small body of work has tried to optimize models to truncate outputs at the point that the model is no longer confident that its prediction will be correct (Foster, 2002; Ueffing and Ney, 2005). Intuitively, such an optimization is a good idea, since deletion of incorrect suffixes is also a time-consuming and cognitively taxing task. However, in practice is has been difficult to design models which can successfully modify prediction length based upon model confidence. In chapter 5, we present some of our own experiments on truncating hypotheses to maximize their utility for translators.

2.4.5 Measuring IMT Performance

As discussed in previous sections, metrics which score IMT systems should ideally be correlated with the amount of human effort that is saved. We can now try to be more specific, and actually discuss and define some of these metrics. In order to measure the quality of an IMT system, we would like to know how much effort the system could save translators, when compared with a baseline such as translation from scratch. Ideally, all systems would be tested in well-designed user studies with human translators; however, because of the difficulty and cost of setting up such studies, in practice IMT systems are often evaluated with *simulated* editing experiments.

In general, any parallel dataset consisting of parallel $(\mathrm{SRC}, \mathrm{REF})$ segments can potentially be used for a simulated IMT experiment. The reference is used as the goal translation, which the IMT system and the translator will collaborate to produce.

Barrachina et al. (2009) use the keystroke-mouse ratio (KSMR) to measure the ratio between the number of user inputs needed when using the IMT system, compared with the number of inputs needed to compose the translation from scratch.

Ueffing and Ney (2005) propose a version of the F1 metric that reinterprets precision and recall for Text Prediction:

$$\mathrm{precision}(p_i, r_i) = \frac{\mathrm{correct}(p_i, r_i)}{\mathrm{length}(p_i)} \tag{2.17}$$

Where $\mathrm{correct}(p_i, r_i)$ is the number of characters in a prediction p_i which are correct according to the reference r_i, and $\mathrm{total}(p_i)$ is the length of the prediction in characters. This term penalizes incorrect suggestions according to their length.

The recall term is a modified version of the keystroke ratio (KSR):

$$\mathrm{recall}(p_i, r_i) = \frac{\mathrm{correct}(p_i, r_i)}{\mathrm{length}(r_i)} \tag{2.18}$$

$\mathrm{length}(r_i)$ returns the number of characters in the reference prediction r_i. This interpretation of recall rewards correct predictions which save the user more operations.

Now we can compute the F1 score of a prediction, and interpret this score as the quality or *relevance* of the prediction (Ueffing and Ney, 2005):

$$\text{score}(p_i, r_i) = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{2.19}$$

Where p_i, and r_i are the system prediction and the reference prediction, respectively. We call this metric **IMT F1**, and use it to evaluate our models in chapter 5.

We now have a metric which can be interpreted as the relevance of a prediction p_i to a query (s_i, t_i), where s_i is the source sentence, and t_i is the existing target prefix. This metric can be used directly to score the output of IMT systems.

2.5 Quality Estimation of MT

Quality estimation (QE) is the task of evaluating a translation system's output without access to reference translations (Specia et al., 2013a). Compared with the long history of MT research, QE is a relatively new area of exploration, and is typically framed as a classification task at the word level and phrase levels, and as a regression task at the sentence and document levels.

Research closely related to QE has a longer history in other areas of NLP, particularly in speech recognition, where confidence estimation (CE) methods are a core part of many systems (Jurafsky and Martin, 2009; Seigel, 2013). Although the first QE methods for MT were motivated by methods from speech recognition, the two tasks are very different, primarily because of the guarantee of left to right decoding with respect to the input in the speech recognition task. MT gives no such guarantee, so the factors influencing whether an output is good or bad are much more diverse in MT.

In the speech recognition domain, *confidence* typically implies that internal information about the decoder is available to guide decisions about output quality. In other words, CE systems are typically designed as *glass box* models, which have access to the same information that the decoder uses to make its decisions, such as language model scores, and phrase table statistics. In contrast, true QE models, as we define them, presume no internal knowledge about the system (or human) that produced the translation, beyond that which can be obtained from a training corpus consisting of parallel $(\text{source}, \text{hypothesis}, \text{reference})$ triples[16].

[16]The workshop on machine translation (WMT) QE tasks release feature sets which do contain internal information about the MT system used to produce translation hypotheses, this is discussed in more detail in chapter 4

QE models can be seen as models which comparing translation system outputs with either abstract or explicit *pseudo-references*. In other words, while metrics such as BLEU, chrF3, and METEOR require at least one gold reference translation to compare with an MT hypothesis, QE models bypass this requirement, but have the same goal of evaluating the quality or correctness of translation. The degree to which a QE model actually builds a literal pseudo-reference depends upon the design of the model. However, the state-of-the-art models described in section 4.3 actually do generate a string representing a guess at the reference translation, then compare this string with the MT hypothesis to obtain quality scores and word-level quality labels.

The remainder of this section provides essential background on QE methods, a discussion of CE methods applied to Interactive Predictive Translation can also be found in chapter 5.

2.5.1 Estimating Translation Quality at Sentence-Level and Word-Level

Translation quality can be estimated at different levels of granularity, ranging from the character level to the document level. Thus far, the most encouraging results have been achieved using sentence-level QE (Bojar et al., 2013, 2014, 2015, 2016, 2017).

Sentence-level QE

Predicting quality at the sentence level allows the use of general features, such as language model scores, features obtained by comparing the parse trees of the source and target (Rubino et al., 2013), and count-based features such as word ratios between source and target hypothesis (Shah et al., 2013). Furthermore, datasets for sentence-level quality estimation are not difficult to construct as a by-product of the normal translation process, because metrics such as human translation edit rate (HTER) (Snover et al., 2006b) and post-editing time (Bojar et al., 2014) are straightforward to compute from training data where each instance is a tuple consisting of: $(\text{source}, \text{hypothesis}, \text{reference})$. Many language service providers (LSPs) have very large datasets of this kind, and may be willing to release some data into the public domain (Bojar et al., 2013, 2014, 2015, 2016). Finally, features computed at the sentence level do not have the same sparsity issues that

word- or phrase-level features may have. Thus, sentence-level QE models can be learned from relatively small datasets, and are likely to generalize reasonably well to unseen data while word-level QE techniques require much larger training datasets.

Word-Level QE

Translation quality prediction at the word-level has only recently achieved a level of reliability that could allow word level QE systems to be useful to translators (Martins et al., 2016; Bojar et al., 2016, 2017). Before 2016, the previous three workshops on machine translation (2013–2015) included a word-level quality estimation task; however, the top performing systems only achieved an F1 score of about 0.4 on the "BAD" class, meaning that the state-of-the-art word level quality estimation still could not identify a large portion of the errors made by an SMT system.

Word-level QE is arguably much difficult than sentence-level QE, and is less well-defined, because complex dependencies exist between the tokens in the target and source sequences – the decision whether a target token is correct or incorrect may hinge upon a combination of information from both sequences, which may or may not be local to the token's position in the target. Furthermore, decisions about a token's quality may depend upon decisions about other nearby tokens, meaning that there are dependencies between the model outputs as well. Lastly, sentence-level quality is likely to be implicit in word-level quality measures, and indeed this has been shown to be the case (Martins et al., 2017; Bojar et al., 2017)[17].

The first two editions of the WMT word level QE tasks used small datasets which were annotated by humans with error types corresponding to the MQM typology[18] (Lommel et al., 2014). The WMT 2015 word level task introduced a new dataset which is much larger than the data provided in previous years, but instead of human annotation, the dataset is automatically tagged with errors using the TER alignment between the SMT hypothesis and post-edited translation. Words in the hypothesis corresponding to the edit operations "Insert" and "Swap/Substitute" are mapped to a "BAD" tag, and all other words are tagged "OK". In the WMT 2015 task, the multi-label tagset from previous WMT editions is thus reduced to binary sequence classification. This automatic tagging of errors

[17] see chapter 7 for a real-world test of this hypothesis
[18] http://www.qt21.eu/mqm-definition/definition-2015-06-16.html

has the advantage that training data is much easier to obtain; however, string alignment based tagging schemes may also introduce noise into the data, and cannot robustly account for deletion and swap operations in the target sequence.

2.5.2 QE and CAT Tools

There has been significant interest in the integration of QE feedback into CAT tools — some research investigates the integration of sentence-level quality estimation into translation interfaces (Turchi et al., 2015; Specia, 2011b); however, exposing automatic quality assessment to translators is still a very new area of research.

As mentioned in section 2.2, translation memories with fuzzy match scoring and thresholding are standard components of all commercial CAT tools. The function of a fuzzy match score is similar to that of a quality score in the sense that it gives the user a rough estimate of the amount of editing that will be required to transform the target side of the TM entry into a correct output. Fuzzy match scores are also typically used with a user-provided threshold — if no entry in the TM is a close enough match for the current segment, the system may decide to query an MT engine, or let the user translate from scratch. If the source and target sides of translation memories are aligned at the word level, the string alignment algorithm used to compute the fuzzy match score can also add visual information about the areas of the target side of the TM that probably will need to be changed, which is itself a form of word-level QE feedback.

The use cases for fuzzy match scoring via string alignment with TM entries are very close to many of the proposed use cases for QE in translation interfaces. The fuzzy match score used with TMs corresponds to a QE system's assessment of the overall quality of a translation hypothesis, while the string alignment information used to guide translators corresponds to labeling quality information at the word level. Esplà-Gomis et al. (2015) presented an interface which used an MT model to provide word-level quality feedback in a CAT tool – in chapter 7, we present a prototype which incorporates word-level QE into a translation interface, and evaluate it in a small user study.

2.5.3 Evaluating Word-Level Quality Estimation Models

The first iterations of the WMT word level QE task used the F1 score of the BAD class as the evaluation metric. However, only considering $\mathrm{F1}_{\mathrm{BAD}}$ can yield deceptive results, because models can directly trade performance on the OK class for performance on the BAD class. Therefore, task organizers switched to the weighted F1 score of the system's predictions (Bojar et al., 2014) as the primary evaluation metric. However, this metric is sensitive to the skew in the class distribution that is typical for QE tasks. Because roughly 20% of the tokens in the training and test datasets are given the BAD label, a model can achieve 80% accuracy by simply labeling every token as OK. Furthermore, the downstream use case for word-level QE is typically considered to be guiding post-editors towards portions of the output which need to be edited, so false negatives on the BAD class are intuitively worse than false negatives on the OK class. In other words, if a portion of a translation is labeled as BAD, but is actually acceptable, it will only create a small amount of extra work for the translator. However, if part of the translation is labeled OK, but is actually BAD, it could directly impact the quality of the resulting translation if the translator trusts the QE model's output.

Therefore, starting in 2016, the WMT task organizers switched to using $\mathrm{F1}_{\mathrm{product}}$ as the primary evaluation metric. This metric weights performance on both classes equally, even though the OK class is much more frequent in the data, achieving a happy medium between overall accuracy and the importance of avoiding false negatives on the BAD class.

2.5.4 Word Level QE Model Types

Structured Prediction for Word Level QE

Most previous work on word level QE has employed the class of ML models known as *structured predictors*. Martins et al. (2016) built a state of the art system using a structured SVMs. Structured predictors such as conditional random fields (SVMs) (Lafferty et al., 2001) output a distribution over structures such as sequences, whereas linear classifiers such as logistic regression can only make individual predictions. There are many choices of algorithms that can learn the feature weights for such models from training data, Mar-

tins et al. (2016) used the max-loss MIRA algorithm (Crammer et al., 2006). In chapter 4 we review the main model types used for word-level QE in more detail.

Feedforward Models

The QUality Estimation from ScraTCH (QUETCH) (Kreutzer et al., 2015b) model is a feed-forward neural network with a single hidden layer that learns bilingual features directly from text. A combination of this model's output with baseline features in an ensemble system achieved the top performance in the WMT 2015 word-level QE task. The model receives each target language word and the corresponding aligned word from the source language as input. To increase the context available for each input token, the words at the left and right of each target and source word are provided as well. The embeddings for all context words are concatenated into a single vector, which is transformed into a probability distribution over the output labels by a feed-forward NN.

The first layer of QUETCH is a word-embedding layer with bias, pre-trained with word2vec (Mikolov et al., 2013), and using a hyperbolic tangent non-linearity. The second layer also uses bias and applies a softmax regularization to estimate the probability of each token being labeled OK or BAD.[19]

Recurrent Models

The use of recurrent neural architectures to model the word-level QE task has been explored by several researchers. Camargo de Souza et al. (2014) use a bidirectional LSTM (Graves, 2013) as a classifier, but do not take advantage of word embeddings, instead using a suite of real-valued features as input. Rubino et al. (2015) use the states of a recurrent neural network to classify target words as GOOD or BAD, class counts are then used to re-rank candidates for automatic post-editing. Recurrent language models have also been used to produce features for sentence-level quality estimation models (Bojar et al., 2015).

2.6 Automatic Post-Editing

Given a source language sentence, and a candidate translation from an unknown provider, we may wish to fix any errors or omissions in the translation candidate automatically,

[19] The softmax function is discussed in detail in chapter 5.

while making as few edits as possible. This task, referred to as, automatic post-editing (APE), has potential applications in most translation workflows. The goal of minimizing the number of edit operations stems from the desire to mimic human translators, who are typically trying to progress through a post-editing task as quickly as possible.

For the purpose of the follow discussion, we will consider datasets consisting of triples with the form $(X, \hat{Y}, Y)$, where $X = x_1 \dots x_T$ is the source sequence, $\hat{Y} = \hat{y}_1 \dots \hat{y}_T$ is an MT hypothesis, and $Y = y_1 \dots y_T$ is the post-edited version of $\hat{Y}$.

The first attempt to create such a system was Simard et al. (2007), who re-purpose a phrase-based SMT system to perform APE by treating the output of another MT system as the "source", and training a new SMT model with $(\hat{Y}, Y)$ pairs. Note that this required no modification of the MT system whatsoever, the only difference from a standard source $\rightarrow$ target MT system is in the semantics of *source* and *target*. Dugast et al. (2007), Isabelle et al. (2007), and Lagarda et al. (2009) proposed similar approaches for statistical post-editing of rule-based MT systems.

The first APE systems did not use any features of the source sequence when training the $\hat{Y} \rightarrow Y$ APE model. To fix this shortcoming, Béchara and van Genabith (2011) propose the creation of a pseudo-language where each source token is actually a concatenation of the original source token x_t with an aligned token from $\hat{Y}$. The alignments are obtained from an automatic word alignment algorithm. Note this is different than an input where every token from $\hat{Y}$ is aligned with a token from the source sequence, because it preserves the order of the original source sequence, and allows un-aligned tokens in $\hat{Y}$ to be dropped from the input. However, this approach is still susceptible to any errors in word alignment, and also sparsifies the training data (Chatterjee et al., 2015).

APE methods have recently shown dramatic improvements due to two innovations:

1. the generation of synthetic training data to augment small APE datasets

2. the use of modified NMT models for APE

These two innovations together have led to a huge jump in APE model performance. In 2015, the first year of the APE shared task at WMT, none of the submitted systems, which were all based upon SMT, were able to outperform a Moses phrase-based source $\rightarrow$

target system (Bojar et al., 2015). In the WMT[20] 2016 APE shared task, the best system outperformed the phrase-based baseline by over five BLEU points, and by over two TER points, by introducing the two innovations above (Junczys-Dowmunt et al., 2016; Bojar et al., 2016). In the 2017 edition of the APE task, all of the best systems were based upon modified NMT models, and used training data augmented with synthetically generated instances (Bojar et al., 2017). Our neural APE system, which was also in the top-ranking systems submitted to WMT 2017, and extends NMT models for APE with new input representations, is discussed in detail in chapter 4.

2.6.1 APE-Based QE Models

Recently, APE-based models have achieved state-of-the-art results in both word level and sentence level quality estimation. These models take a radically different approach from feature based models, directly predicting the corrected translation as output. The corrected translation can then be treated as a pseudo-reference, and aligned with the original MT hypothesis to obtain word-level QE labels, corresponding to string edit operations. Such models are discussed in detail in section 4.3.

2.7 Discussion

We have now given the necessary background for the rest of the thesis. Individual chapters include additional background where relevant, while making reference to this chapter where possible to avoid redundancy. This chapter has necessarily omitted much relevant work for the sake of keeping our discussion reasonably brief, but we hope the reader will refer to the original cited sources where more information on a certain topic is desired.

[20]Note that the *workshop* on machine translation officially changed its name to the *conference* on machine translation in 2016. In this thesis, we use the acronym WMT for all sessions of this event

Chapter 3

HandyCAT: A Platform for Translation Process Research

This chapter discusses the design and implementation of a computer-aided translation (CAT) interface for translation process research. HandyCAT is our open-source[1] implementation of a flexible web-based CAT tool (Lewis et al. 2014), specifically designed with inter-operability and extensibility in mind, following the core principles and design considerations laid out in section 3.2. We are focused upon *RQ1* throughout this chapter, creating an environment that enables testing new translation technologies in CAT interfaces.

Throughout this chapter, we use the term *component* to mean a part of a user interface which encapsulates a high-level functionality. For example, we consider the part of the interface that allows the user to interact with a source sequence and its corresponding translation to be a *segment-level* component. In this abstract view, components are very similar to classes in object-oriented programming, which provide an effective means of hiding complexity, while allowing well-defined communication with the world via a contract called an *interface*.[2]

The chapter is organized as follows: section 3.1 motivates our case for designing and implementing a new CAT platform for research. Section 3.2 introduces *component-centric design*, our design methodology which proposes a simple type system for CAT

[1] https://github.com/chrishokamp/handycat

[2] A note on terminology – throughout this chapter the term *interface* is used in two ways: (1) as in *User Interface* (UI), and (2) as the formal definition of the functions and data exposed by a component, the same as the definition of *interface* or *trait* in object-oriented programming.

components, and Section 3.3 lays out the design requirements for the translation interface. Section 3.4 discusses the components built for the various translation tasks introduced in section 2.1. Finally, sections 3.6 and 3.7, presents two user studies conducted in collaboration with translation studies researchers, which tested new components in HandyCAT. The results of these user studies motivated much of the research discussed in the remaining chapters.

3.1 Motivation

As discussed in section 2.2, the requirements of interfaces for research are quite different from those for interfaces designed to facilitate translators' day-to-day work. CAT tools, like most desktop-based user interfaces (UIs), are typically designed as self-contained platforms which attempt to implement all important features within a monolithic framework. Although many existing CAT tools provide a full suite of features which satisfy all of the core user needs, this design methodology is not conducive to research into novel UI components, because the implementation of new functionality necessitates significant modification of the platform source code, which may be impossible or very difficult in practice.

Furthermore, many components rely upon interaction with one or more data services, each of which may require significant computational resources and storage resources not available on a user's local device. This means that CAT tool developers must either implement a universal interface for components to connect to remote services, or allow each component to specify the services it requires, assuming that the logic for interacting with the service is contained within the component itself. Examples of data services that may be implemented as remote services include machine translation, translation memories, glossaries, and concordancers.

A final point for consideration is the potential for re-usability of the elements in CAT tools. Because the functionalities expected by users of translation interfaces are very well-defined when compared with the myriad possibilities for functionality related to arbitrary text composition tasks, there is a great potential for component re-usability within translation interfaces. Furthermore, many of the standard elements in CAT tools, such as

glossaries and TMs, are arguably useful both as standalone applications, and as integrated components. Ideally, these elements should be able to fulfill both purposes with little or no modification.

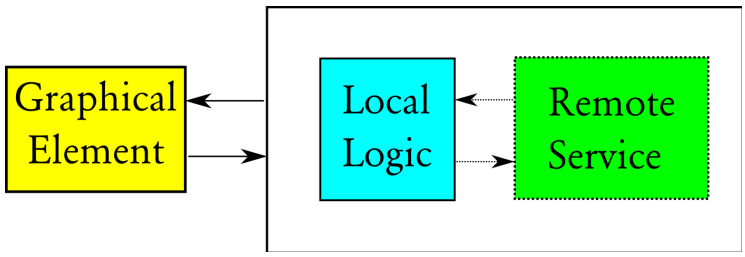


Figure 3.1: An individual component, consisting of a graphical element, local client-side logic, and an optional interface to a remote microservice.

Designing CAT tools as monolithic, self-contained interfaces thus requires developers to make many assumptions to be made about what kind of components may be implemented, and greatly limits the flexibility of the interface. An alternative approach is a *microservice*-based architecture (Newman, 2015), where each component can be modeled as a mini-UI of its own, exposing an interface for communicating with the rest of the tool via the type system defined by the designer. By *type system* we mean a hierarchical structure of classes as could be created with an object-oriented programming language, enabling inheritance and allowing a distinction between public and private data for each class. Figure 3.1 visualizes a single component. The microservice-based approach views the application as an amalgamation of "sub-applications" which can communicate with each other by conforming to the contracts imposed by the type system. We call this approach component-centric design (CCD), and HandyCAT was implemented according to this paradigm.

3.1.1 Web Based CAT Tools

Modern web browsers provide a flexible, high-level, mostly standardized application programming interface (API) that is the ideal platform to implement the desired customizability of a CAT tool for research. The HTML5 specification (Danilo et al., 2017) in particular has provided a complete suite of technologies suitable for implementing almost any kind of graphical application. The web browser is also ubiquitous on all modern personal computing devices and all operating systems. In comparison with other high-level

Pros	Cons
Fast development iterations	Performance limitations due to web browser and ne
Easy to implement new graphical components	Difficult to debug
Easy to connect with remote services	Limited access to local resources

Table 3.1: Some pros and cons of web-based applications

frameworks for designing user-interfaces, such as the Swing libraries for the Java programming language (Eckstein et al., 1998), web browsers provide a higher-level API for designing graphical components with HTML and CSS, as well as Javascript, a powerful scripting language. Javascript has the additional benefit that it is very well suited to asynchronous applications like user interfaces. Table 3.1 lists some additional advantages and disadvantages of web-based applications.

Particularly in the context of CAT research, web interfaces have the additional advantage that they do not need to be installed on a local machine – simply sending the test subject a URL together with a user name and password is enough, assuming they have access to a web browser. This streamlines the setup workflow, significantly lowering the barrier to conducting user studies.

Finally, the emergence of modern UI frameworks targeted at web browsers has revolutionized application design over the past 5-10 years. Beginning with libraries such as Bootstrap[3], and progressing through lightweight frameworks such as BackboneJS[4], modern client-side Javascript frameworks such as AngularJS[5], Ember.js[6], React[7], Meteor[8], and Vue.js[9] have made web applications fast to develop and easy to maintain.

3.1.2 To extend or to build from scratch?

As discussed in section 2.2, several open-source CAT tools already existed when this work was beginning. We carefully evaluated each of these as possible candidates for extension. Extending an existing tool would have been the ideal way to test new interface functionality, as it would have saved much work involved with building the basic interface, and implementing functionalities such as user accounts, logging, and persistence, which

[3] https://getbootstrap.com/
[4] http://backbonejs.org/
[5] https://angularjs.org/
[6] https://www.emberjs.com/
[7] https://reactjs.org/
[8] https://www.meteor.com/
[9] https://vuejs.org/

are necessary, but not directly related to the goals of this thesis.

Of the CAT UIs discussed in section 2.2, **CasmaCAT** and **OmegaT** were our two initial candidates for extensible, open-source platforms for CAT tool research. However, each had important disadvantages that led us to forgo extending it, in favor of implementing our own tool from the ground up.

CasmaCAT is designed as a monolithic tool with a PHP backend. Although PHP is a historically important and popular language for developing server-side web applications, the server-side design makes implementing new user interface components challenging, when compared with client-side frameworks such as AngularJS. The popularity of PHP was also waning when this project was beginning, and the CasmaCAT project was drawing to a close, casting the long-term maintainability of the interface into doubt.

OmegaT is a Java-based desktop CAT tool. OmegaT is a fully-featured CAT tool which can be used in professional translation workflows, but is not easy to extend, particularly because of its idiosyncratic data model (Moran et al., 2014). However, the primary factor that led us to disqualify OmegaT was our desire to use a web-based tool, both because of the ease of development discussed above, and because of the ease of conducting user studies remotely with a web-based interface, which simplifies and lowers the cost of user studies for both researchers and study participants.

We therefore decided to proceed with developing our own CAT tool for research, which is now called HandyCAT. The following sections discuss our design and implementation methodologies in detail.

3.2 Design

The purpose of a CAT tool is to facilitate modification of the state of the *document data model*. The translation data model is a hierarchical, object-oriented view of the translation data, which should be serializable into XML, JSON, or another structured representation. Because translation tasks are a relatively well-defined subset of text editing tasks, several attempts have been made to design a universal document specification for translation data exchange. HandyCAT supports the XLIFF 1.2 and 2.0 data models (Schnabel et al., 2015; Tingley, 2015), and the main HandyCAT component areas correspond to XLIFF elements

Interactive Element	Description
Segment Area	Encapsulates a source segment and the corresponding translation
Data Service	**Description**
Translation Memory	Searches for full, fuzzy, or partial matches for the source segment in a database of $(\mathrm{source}, \mathrm{target})$ pairs

Table 3.2: Examples of interactive elements and data services in CAT tools

(see section 3.2.3).

Translation interfaces update an mutable internal representation of the translation data model as the user translates. This internal representation can then be mapped into a platform-agnostic format such as XLIFF once the translation job is finished (Lewis et al., 2014). User modification of the data model is accomplished via interactions facilitated by components. The following section discusses our system for describing and formalizing user interface components in CAT tools.

3.2.1 Component-Centric Design: A Type System for CAT Components

CCD is a framework for designing elements of a CAT tool as embedded applications, which are connected and composed together via well-defined interfaces. A component is a means of transforming and/or rendering some data to the end user, optionally with interactive capabilities which allow users to modify the underlying data model. Components in translation interfaces may be composed from of three types of objects: (1) *data services*, services that transform textual input data, (2) *interactive elements*, which provide the means for users to view and possibly modify parts of the data model, and (3) other components. In other words, a component can be built from other sub-components, interactive UI elements and data services.

A complete translation interface is thus a hierarchy of data services and interactive elements, which can be represented as a directed graph. This abstract view of the interface allows dynamic configuration based on user needs, and simplifies the design and integration of new components.

3.2.2 Formalizing Component-Centric Design

Now that we have given examples of some component types for CAT interfaces, we can give a formal specification of an *Interface*. An Interface is a set of one or more components $\{\mathbb{C}\}$, where each $C \in \{\mathbb{C}\}$ can be written as a tuple $\langle E, D*, C* \rangle$, and:

E = Interactive Element

D = Data Service

C = Component

* = zero or more

Some previous work, e.g. Green et al. (2014b), has argued that the distinction between data services and graphical elements is not useful in practice, because the functions which provide data to an interface are tightly coupled to the graphical elements which allow users to interact with the data. Although some integration between data services and graphical elements will of course be necessary in order to create a useful interface, ideally the "integration layer" should be kept to the absolute minimum, simplifying the design of new elements, and maximizing their re-usability in other contexts. Components should thus follow common object-oriented design principles, especially when defining the interfaces with data services.

Generally, graphical interface components can be decoupled from the data services they rely on to some extent, but each component will inevitably have some tight integration with the data service(s) it relies upon. Thus, the interface between graphical component and services must be carefully designed. Along the same lines, the interface between user and component should be encapsulated from the supporting data services as much as possible.

3.2.3 Document Data Model

The XLIFF 1.2 and 2.0 standards (Schnabel et al., 2015; Tingley, 2015) provide an abstract, object-oriented view of translation data. The most important object in XLIFF documents is the *translation-unit* or *unit* (`<trans-unit>` in XLIFF 1.2 and `<unit>` in

XLIFF 2.0). Figure 3.2 shows an example of a XLIFF unit element containing two segments. HandyCAT was designed to support the basic elements of both the XLIFF 1.2 and 2.0 standards, allowing users to upload XLIFF documents and edit them inside HandyCAT, and to export projects as XLIFF 2.0 documents[10].

```
<unit id="1">
   <segment>
       <source>First sentence.</source>
       <target>Première phrase.</target>
   </segment>
   <segment>
       <source>Second sentence.</source>
   </segment>
</unit>
```

Figure 3.2: An example translation unit in an XLIFF 2.0 DOM.

HandyCAT implements a client-side XLIFF document parser in Javascript, and directly writes changes to the data model into a stateful XLIFF document object model (DOM) using the HTML 5 DOM API[11], which is supported by all modern web browsers. To the best of our knowledge, this is the only CAT tool implementation which directly uses the XLIFF DOM as a data model, instead of transforming to and from XLIFF from an implementation-specific data model. This one-to-one correspondence between the serialization format (XLIFF XML), and the XLIFF DOM is a powerful and simple way to approach interaction with translation documents, and proceeds directly from our decision to use the modern web browser as our implementation platform.

Although the (im-)mutability of certain element types is not enforced by the XLIFF standard, in HandyCAT, all elements in the XLIFF DOM are immutable, except for the `<target></target>` elements inside `<segment></segment>` elements. This corresponds to the purpose of CAT tools as we have already defined it: to allow the user to produce or modify target language translations of source language segments.

[10] We additionally released a library for creating XLIFF documents as a standalone project at `https://github.com/chrishokamp/node-xliff`

[11] `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model`

3.3 Requirements

We have now proposed a flexible design framework for CAT tools, and described a data model that can be used to store and manipulate the state of the interface as users complete translation tasks. The following section discusses the design requirements of CAT interfaces, enumerating the fundamental components in CAT tools, and then discusses how these components have been implemented in HandyCAT.

The ability to view a source segment and to compose and persist (save) a corresponding target segment are the baseline functionalities required for a translation interface. Most existing CAT tools provide a set of standard interactions, such as editing of target segments, confirming segments, and searching a glossary, within a segment-oriented view. The additional features provided by a CAT tool are either focused upon providing translators with metadata to facilitate the translation process, such as automatic machine translation and translation memory matches, or designed to assist users in other stages of the localisation workflow, such as document conversion, handling markup and tags, or terminology extraction.

The core functions implemented by a CAT tool include displaying the the source segment, and allowing the user to edit a target sentence, either by typing from scratch, or by post-editing a TM segment or an MT hypothesis. By abstracting over the common graphical affordances (Greeno, 1994) in CAT interfaces, we identify a set of *functional areas*, which provide an abstract representation of the graphical areas that render translation data to the user. Any translation interface will include areas dedicated to these functions, but the interactions implemented by the components within the functional areas may be drastically different from implementation to implementation.

The core set of containers common to all CAT tools with a two-dimensional graphical interface includes the source area, the target area, and the tooling area. Note that parameters which control the graphical rendering of the areas, i.e. their position and size on the page, are hyperparameters of the application, and can be determined by tuning/optimization, trial and error, or constraints imposed by the user's device. On devices with smaller form factors, such as mobile devices, it is likely that most areas will be hidden

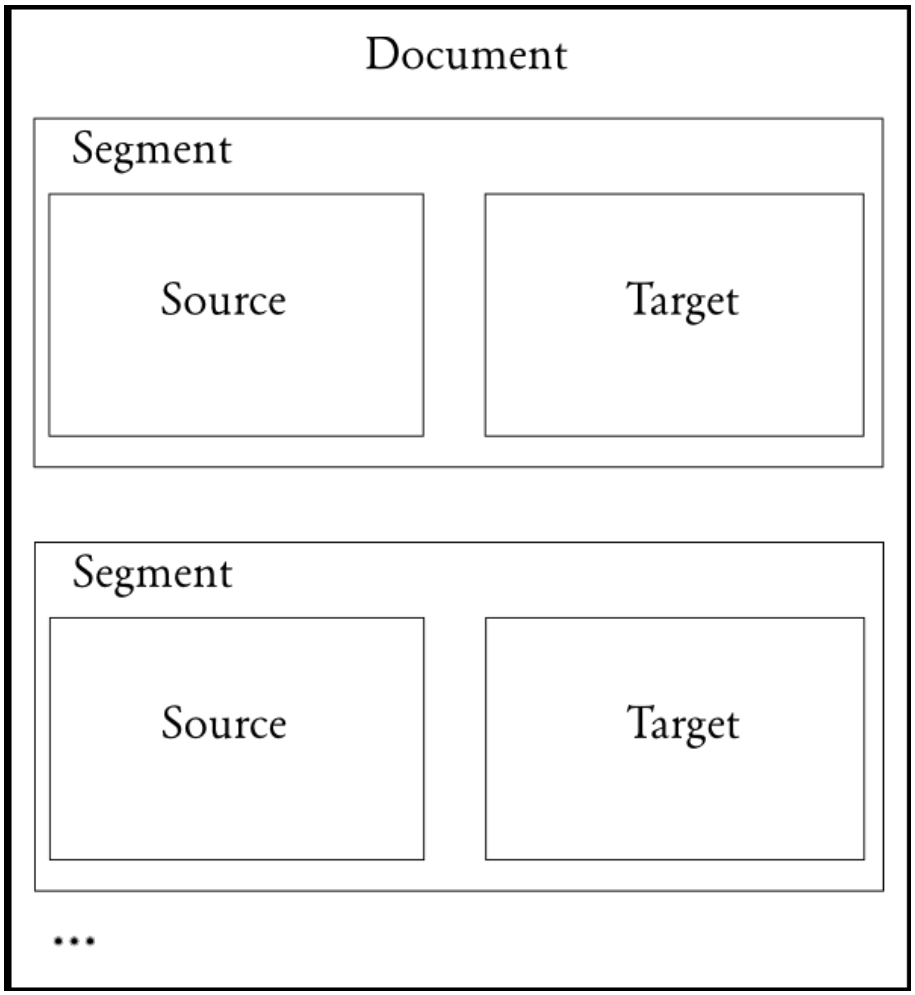
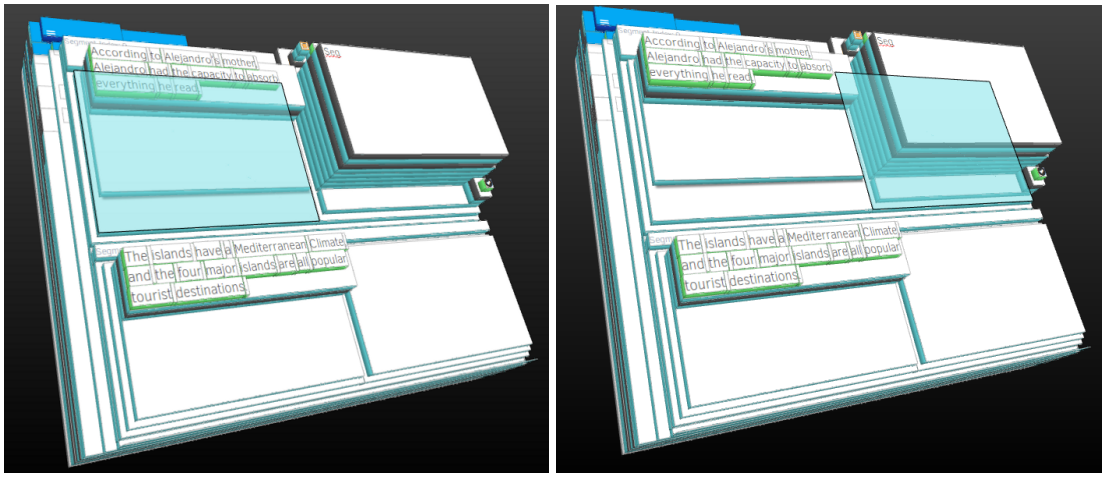


Figure 3.3: Top level abstract component areas



(a) Source Area (b) Target Area

Figure 3.4: The source and target areas in HandyCAT, highlighted in green. These renderings are produced by the three-dimensional DOM visualizer available in Mozilla Firefox, and emphasize the nested component areas in HandyCAT.

by default, and accessed via specific interactions, whereas devices with more display area may display all areas simultaneously.

Component Interfaces
Document
Segment
Source
Target
Toolbar

Table 3.3: Examples of CAT component interfaces

HandyCAT provides several predefined functional areas which have access to a specific part of the data model or document tree. Components and the stateful objects that store their data also correspond roughly to XLIFF elements; thus, there is a three-way correspondence between graphical components, the HandyCAT data model, and the XLIFF DOM representing the translation document. Table 3.3 lists some of the main component interfaces that will be present in any CAT tool.

Developers can specify which parts of the data model their component can access by placing in inside a container, such as the *EditArea*, or the *TargetArea*. By defining the component at the right level in the hierarchy, developers can ensure that their components are isolated, and that the document model is always synchronized with UI state. Figure 3.4 visualizes the container hierarchy in HandyCAT.

Tooling components are "floating" or "singleton" components – there is typically only one instance of a tooling component, and it may render in different areas of the interface depending upon the user's context. The toolbar in HandyCAT is an example of such a component. A single toolbar instance moves below the currently active segment, acting as a container for *toolbar components*, which implement functionality such as translation memory lookup, searching in knowledge bases, or named entity annotation (Hokamp, 2015). In addition to the document model, *tooling* components also have access to interface state, so they can respond to changes such as segments being activated or deactivated, or listen for arbitrary external events, such as a word being clicked in the source or target area.

3.4 Component Implementations

Translation process research on explicit interaction in CAT tools focuses upon the design and analysis of systems where users interact with one or more data services via an enhanced editing component, which could be a drop-down or pop-over element placed near the user's cursor in an editing area (Foster, 2002; Barrachina et al., 2009; Green, 2014), or a "typeahead" style component (see sections 3.6 and 3.7). However, other graphical components and interaction modalities beyond autocompletion have been proposed (Koehn, 2009b; Alabau et al., 2011).

As discussed in section 3.3, HandyCAT defines graphical areas which are tightly coupled with interfaces to the translation data model. Although this design is appealing in the abstract, without concrete component implementations, the feature is meaningless. This section briefly discusses several actual component implementations in HandyCAT, the majority of which are components for modifying the target text of segments in the translation data model.

Segment area components

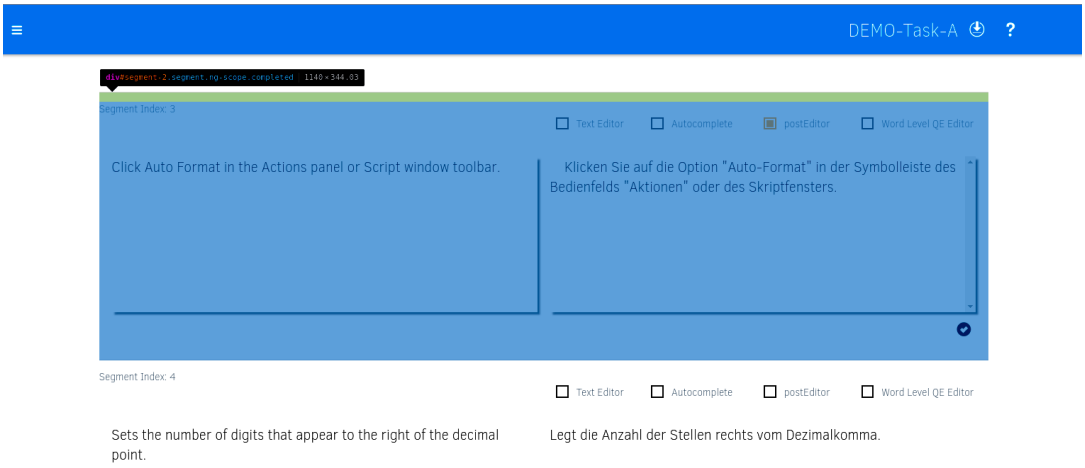


Figure 3.5: The graphical element corresponding to segment area components

Segment area components (Figure 3.5) are wrappers for a single segment, consisting of a $(\text{source}, \text{target})$ pair. This component type corresponds to the <unit> XLIFF 2.0 selector (<trans−unit> in XLIFF 1.2). The component is responsible for rendering segment-

level information, such as the segment's index in the project, as well as for configuring the available sub-components in the source and target areas, and for storing segment-level information such as alignment information about the source and target. The *display* of the source and target parts of a segment is also controlled by this component; for example, the configuration of the segment area component determines whether the target segment will render to the right of the source segment, or below it.

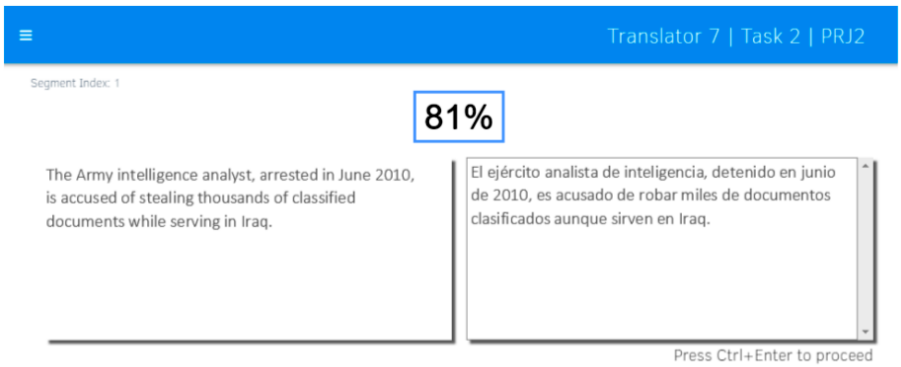


Figure 3.6: A segment area component that renders a segment-level quality score which the user must click before she can edit the target hypothesis.

At least one research project (Teixeira and O'Brien, 2017) has designed a specialized segment area component for HandyCAT, which includes a sentence level quality score above each segment. This work used eye tracking to determine the impact of showing quality scores to translators before they begin post-editing each segment (see figure 3.6).

Source-area components

Beyond simple rendering of the source segment, source-area components may add interaction features which allow users to perform operations such as looking up a selected substring in a glossary, translation memory, or concordancer (see figure 3.9). For example, Hokamp (2015) presents a source component which has access to an entity-linking API, enabling the automatic recognition of named entities in the source segment, and suggesting translations for these entities by leveraging the multilingual links in DBpedia.(Auer et al., 2007; Daiber et al., 2013) Figure 3.7 shows the interface element corresponding to source-area components.

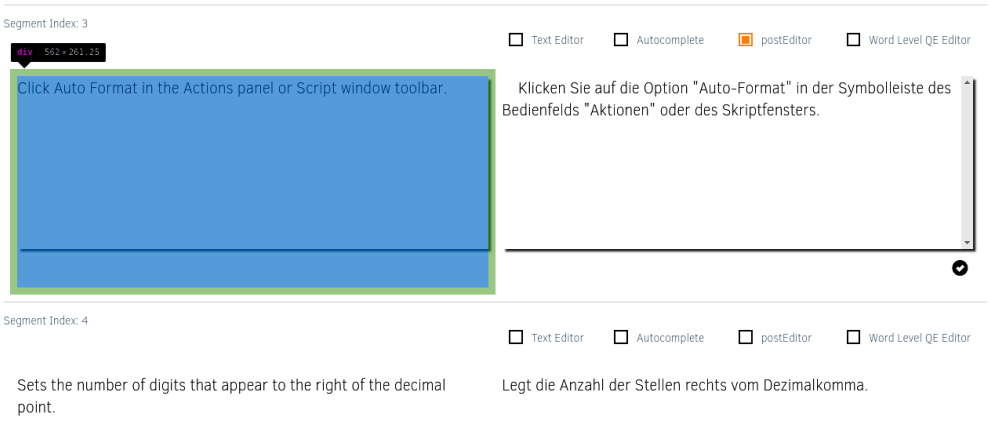


Figure 3.7: The graphical element corresponding to source-area components

Target-area components

Because target area components are the means by which translators can actually modify the text of a translation, this class of components has the most alternatives of any component type in HandyCAT. Most of the components implementing special functionality were created for specific research projects. Table 3.4 describes some of the target area component implementations currently available in HandyCAT.

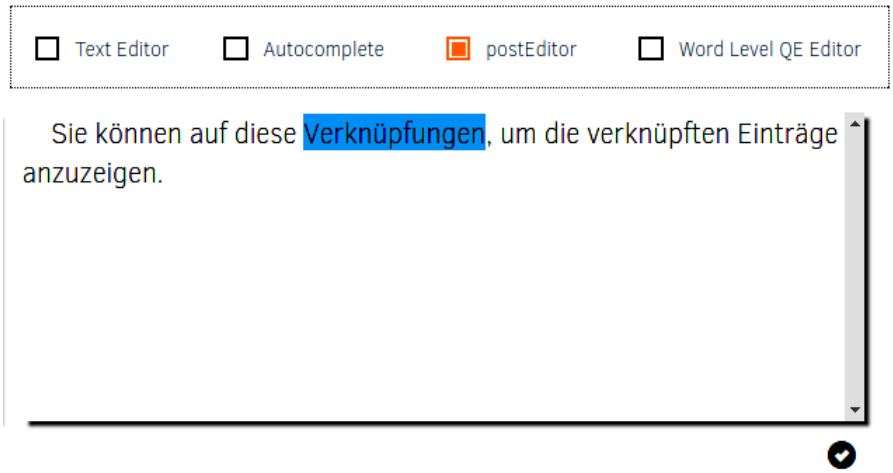


Figure 3.8: Multiple target components on the same segment – user can switch between components using the selector.

In the target area of the interface, the *component selector* enables translators to use multiple components on the same segment (see figure 3.8). For example, the translation resource selector can be used to chose the best output from several MT systems, then the

Component Name	Description
Plaintext Editor	Allows translation from scratch in a standard HTML 5 text-editing element.
Post-Editor	Allows user to perform high-level editing operations on spans of text by selecting text and interacting with a drop-down menu of actions.
Word-Level QE Editor	Renders realtime word-level quality information about words in the target translation using a quality estimation server.
Typeahead Editor	suggests completions for the text in the target segment using one or more autocompletion services.
Interactive Post-Editor	extends the Post-Editor by allowing the user to confirm spans of text, which are maintained when new translations are requested.
Translation Resource Selector	Allows the user to choose a translation out of several options provided by different translation services.

Table 3.4: Target-area components implemented in HandyCAT

post-editor can be used to make any changes to the chosen segment.

Toolbar-area components

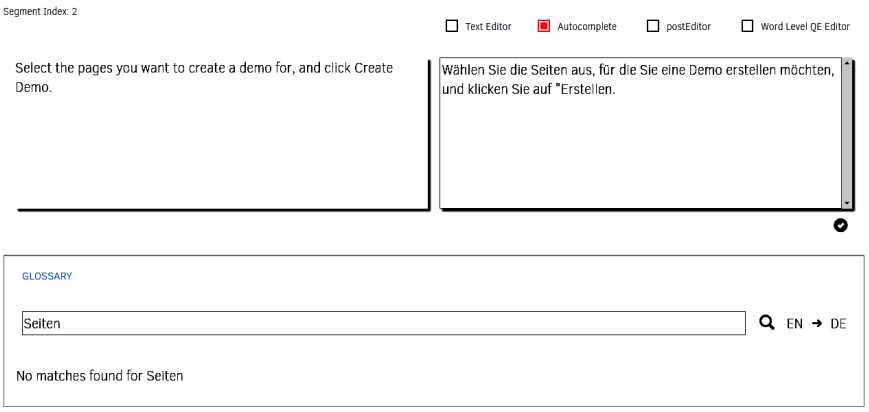


Figure 3.9: The singleton element corresponding to the toolbar

The toolbar area is unique in that it is a singleton element (i.e. the interface only contains a single instance of the toolbar) that renders in the interface according to the current active segment. Toolbar components have access to the entire translation data model, but are typically used to provide data about the currently active segment. In HandyCAT, toolbar components include glossaries, thesauri, and concordancers. Note that toolbar components are typically tightly coupled to a remote microservice (see section 3.5). Figure 3.9 shows the toolbar area below the active segment, containing a simple glossary component.

Data Service	Description	Used by Component
Trie Autocomplete	A REST server to a trie-based data-structure for fast autocompletion based on word lists	Typeahead Editor
Vocabulary Server	A REST server to dictionaries mapping source words to target words	Typeahead Editor
Language Model Scorer	A REST server to SRILM language models	Typeahead Editor
Interactive MT server	A wrapper around IMT models built with our Neural IMT framework	Typeahead Editor
Glossary	A service which takes words as queries and provides their definitions and synonyms via the Glosbe API	Glossary
Translation Memory	Provides exact and fuzzy matches for source and target segments	Translation Resource Provider
Word Level QE server	Annotates spans of text with word-level quality scores	Interactive Post-Editor
Constrained Decoding server	Allows translation conforming to user-provided constraints	Interactive Post-Editor

Table 3.5: Examples of Data Services used by HandyCAT components – all of these services are our own implementations. Some are located within HandyCAT itself, others are standalone projects that can be plugged into HandyCAT

3.5 Services and Microservices

Many components require one or more data services services to function. For example, components which provide auto-complete functionality may use an auto-completion service, or an IMT system to produce the completions that are provided to the user in the interface. In HandyCAT, data services are implemented as *microservices*. Developers are free to implement whatever services their components may need, table 3.5 lists some of the data services implemented over the course of this thesis, and points to the component(s) that make use of the each service.

We have now described the core design principles and implementation of HandyCAT. The following two sections describe pilot user studies conducted using this interface.

3.6 A Pilot Study On Text Prediction Using SMT Phrase Tables

Autocompletion is arguably the most successful user-facing application of language modeling in widespread use today. Autocomplete functionality is ubiquitous across modern text interfaces. Especially for mobile devices, autocompletion utilities are an indispensable part of the user experience, because of the dramatic reduction in typing speed with caused by basic touchscreen text-entry interfaces.

Despite the pervasiveness of autocompletion in content creation workflows, there is very little research evaluating autocompletion within the context of CAT. There are two major benefits to autocompletion when compared with composition from scratch: (1) autocompletion may help the user to find better ways to translate the source segment by providing a variety of suggestions, and (2) autocompletion may save the user time by minimizing the number of input operations needed to complete a translation. In the following experiments, we focus on (2), attempting to design a system which helps users to complete a translation task more quickly.

Evaluating Autocompletion for CAT

The following pilot study evaluated a simple but effective approach to autocompletion or "type-ahead" components based on elements of a traditional statistical machine translation system. We compare two auto-completion engines in a user study with 16 translation students who are native Spanish speakers. The *target auto-completer* uses prefix matching over the known vocabulary of the target language. This mimics the auto-completion utilities typically found in smartphones or word-processing software. The *source-constrained auto-completer* leverages an SMT phrase table and a target-side language model to provide enhanced suggestions to translators, by using the source sentence as an additional context for filtering suggestions. When a translator starts translating a segment, the possible completions are first retrieved from a data service which queries a Moses phrase table built from the WMT 2013 news translation task's English-Spanish dataset. As the user translates, the system dynamically re-ranks the completion candidates using a language model conditioned on the current target prefix.

When compared to autocompletion strategies using interactive machine translation (Green et al., 2014b; Bender et al., 2005; Ueffing and Ney, 2005; Koehn, 2009b; Ortiz-Martinez, 2011), this approach has the advantage that it can be implemented without deploying a full-fledged machine translation system. Phrase-table based autocompletion also has much smaller computational requirements than an IMT system, which must perform online prefix-decoding (see section 2.4).

As discussed in section 2.4.1, the task of text-prediction can be framed as the search for the best continuation $\hat{c}$, given a prefix (equation 2.10). The source-constrained auto-

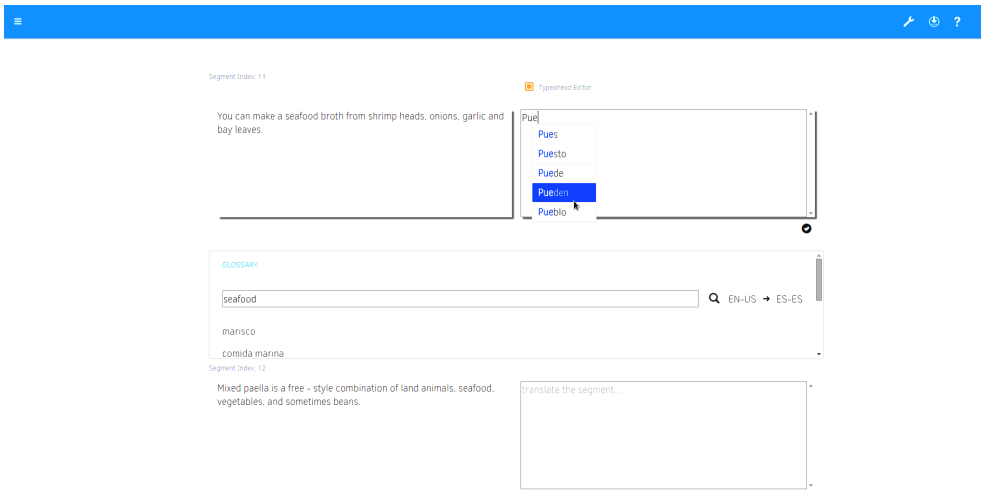


Figure 3.10: A target-side auto-complete component backed by a phrase table and language model.

completer used in these experiments is described by equation 3.1:

$$\hat{c} = \underset{c \in C_{\text{source}}}{\operatorname{argmax}} P(c|\text{prefix}, \text{source}) \tag{3.1}$$

Where C_{source} indicates the set of of all phrases returned by the phrase table for a particular source sentence[12]

Although IMT via tight coupling with the SMT decoder might provide better suggestions by directly leveraging all features implemented in the SMT decoder, such systems can be too slow for real-time use without significant tuning (Green, 2014), because the computational requirements of the decoder are very demanding. Stack-based SMT decoders generally have a trade-off between computational requirements and quality, which can be controlled by the hyper-parameters of the system, including the stack size and the maximum n-gram size of the language model.

The system implemented for this pilot study effectively uses a target-side language model as the only feature in a prefix decoder, while filtering candidate completions using an SMT phrase table. The user can quickly filter the phrase options by typing a prefix of the desired word or phrase. Despite this simplification, we observe a significant improvement in translation speed when compared to a baseline prefix autocompletion engine.

[12]The phrase table is queried with every possible n-gram in the source sentence up to size=3.

3.6.1 Experimental Design

Figure 3.10 shows our implementation of the autocompletion component used for the experiments discussed in this section. Because the target auto-completer and the source-constrained auto-completer only differ in the way their suggestions are generated, we can use the same graphical component in both settings.

The baseline for our experiments is a monolingual auto-completer which has no knowledge of the source segment being translated. The word dictionary used by this auto-completer was created by sorting the words in the Spanish portion of the News-Commentary corpus (Tiedemann, 2012) by frequency. Although this engine is insensitive to the source, it can arguably still make translators substantially faster, by saving them keystrokes in the same manner that autocompletion is used in other monolingual text-editing interfaces.

When the source-constrained auto-completer is in use, before the user has typed anything, the phrases C_{source} are simply ranked according to $P_{\text{LM}}(c|\text{START})$, the probability that the phrase starts a sentence according to the language model. As the user types a word, the phrases in C_{source} are filtered to those which begin with the user's current prefix. When the spacebar is pressed, the phrases in C_{source} are again re-ranked according to $P_{\text{LM}}(s|\hat{y}_0 \dots \hat{y}_{t-1})$, where $\hat{y}_0 \dots \hat{y}_{t-1}$ is the current word-level prefix already entered by the user. In the case that the user has entered a string of characters that does not match the beginning of any phrase in C_{source}, the source-constrained auto-completer falls back to the same dictionary of words used by the baseline component.

To create the test projects for the user study, we selected 30 sentences from the English Wikipedia page for "Malaga"[13], which were divided into two datasets of 15 sentences each. Two XLIFF files with 15 source segments each, and empty target segments were then created from the extracted sentences. The files were balanced for number of words and difficulty of translation by a professional translator. Because there are two possible orderings of the translation tasks, and two autocompletion utilities to test, we created eight experimental groups which account for every permutation of task ordering and component configuration. Participants were each randomly assigned to one of the eight groups.

[13] https://en.wikipedia.org/wiki/Malaga

3.6.2 Results

Although we collected data from about 30 translators, many sessions were incomplete, or users had made significant errors, such as neglecting to mark translated segments as complete. Filtering the data to remove these corrupted sessions resulted in 16 clean sessions – the following analysis uses data from these sessions only.

In order to account for possible mistakes during the translation process, such as late confirmation of a segment, we also discard the worst outliers for each test segment.

Dataset	**Avg. Time (seconds)**
A	79.53
B	76.47

Table 3.6: Average sentence completion time for each dataset

Autocomplete Type	**Avg. Time (seconds)**
Default	82.75
PT-backed	**73.25**

Table 3.7: Average sentence completion time for each autocomplete type

Table 3.6 shows that the average completion times for each sentence in both datasets were quite similar. We interpret this to mean that the translation difficulty of the sentences in the two datasets was comparable.

Table 3.7 shows the average completion time for a segment using the two autocomplete types. The PT-backed autocomplete allows translators to complete a segment more than 9 seconds faster on average — a statistically significant difference with a p-value of 0.011 when measured with Student's T-test.

Figure 3.11 presents the per-sentence average times for each sentence in each project. In this figure, the difference between the average editing time for the two autocomplete settings is visualized by the green bars. This figure gives a visual confirmation that the source-constrained autocomplete made translators faster on the majority of segments.

According to the our experimental results, translators are more than 10% faster with the enhanced autocomplete. This is an encouraging finding, especially given the relative simplicity of the phrase-table and LM-backed autocompletion components.

Through this pilot study evaluating two implementations of text prediction within a

Figure 3.11: Average time per segment for each autocomplete type. Segments are sorted by source length (ascending).

CAT tool, we established the general utility of autocompletion in translation interfaces. We additionally confirmed the stability and flexibility of HandyCAT as a platform for translation process research, confirming that the tool could be used in settings with up to 30 concurrent users.

3.7 A Pilot Study on Post-Editing Actions

Despite the increasing use of post-editing as a core part of professional translation workflows, there is very little work on new user interfaces specifically designed for the post-editing task. Some of the high-level research goals for post-editing interfaces are to:

1. Understand how translators use current text-editing interfaces to perform post-editing
2. Discover useful *macro* operations which encapsulate several fine-grained interactions
3. Design and test interfaces that implement functionality specifically for post-editing

One useful abstraction of the editing process is as a sequence of basic *edit-operations*, which are classically described as *insertion*, *deletion*, *replacement*, and *substitution* (Ristad and Yianilos, 1998; Snover et al., 2009). Although these coarse operations may not correspond directly to the segment- or text-level cognitive processes that occur in a translator's brain as they post-edit, they are still useful "macro" operations that may yield a deeper understanding of the editing process than raw keystroke logs, or character-level diffs.

Taking the basic edit operations as inspiration, in collaboration with Félix do Carmo at the University of Porto, we designed and implemented a component providing a graphical interface to these four editing actions, along with interaction flows that encapsulate each of these basic operations. This component was then tested in a large-scale user study with approximately 60 translators.

The main purpose of this study was to test a new interactive modality purpose-designed for post-editing, where translators have a limited number of "macro" actions available, corresponding to the string operations INSERT, DELETE, MOVE, and REPLACE.

By limiting translators to these actions, the sequences of keystrokes in the HandyCAT session logs are directly transformable into semantic chunks, which we hope correspond to specific user intents in the course of post editing. For example, when a user enters replace mode, the purpose of the sequence of keystrokes until the mode is exited is known to be the replacement of the originally selected phrase. Raw events in the interface are thereby closely tied to easily understandable intents, facilitating analysis of the post-editing task.

Post-Editing Component Design and Implementation

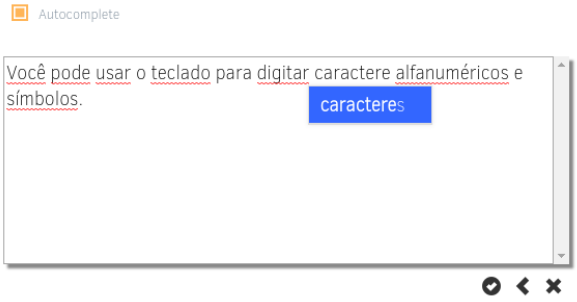
The post-editing component is implemented as target-side text area, where users must interact with the target hypothesis by highlighting text, choosing one of the editing actions or *modes*, and then using the mode's functionality to edit the text. Each editing mode is exited by pressing the ESCAPE key. Table 3.8 describes the purpose of each of the editing modes. Note that all possible editing operations could actually be accomplished with DELETE + INSERT alone, thus MOVE and REPLACE are effectively convenience modes which allow the user to perform specific macro operations.

Name	**Description**	**Purpose**
INSERT	A cursor is placed at the end of the highlighted word or whitespace, prompting the user to enter new text.	Add text that is missing from the translation hypothesis.
DELETE	Deletes the highlighted word.	Remove incorrect text.
MOVE	After highlighting a word, the user selects another location in the text where the word should be moved.	Fix word-order issues in the translation.
REPLACE	The highlighted word or whitespace is deleted and replaced with a text cursor, prompting the user to enter new text.	Shortcut for DELETE followed by INSERT.

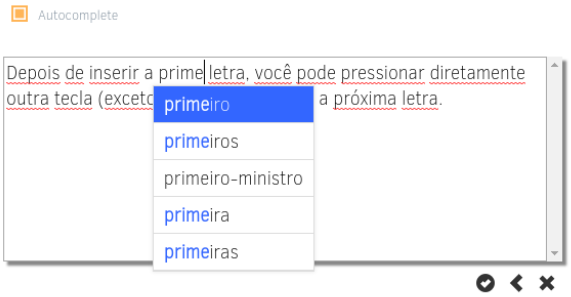
Table 3.8: Descriptions of the four editing modes in the post-editing actions user study.

The implementation of the above editing mode specifications as robust components in HandyCAT was a non-trivial undertaking. The design specifications provided by Félix do Carmo outlined the high-level desired functionality from the components, but did not constitute a full design specification, and many design iterations were needed before the

components were ready to be tested.



(a) Typeahead component screenshot one.



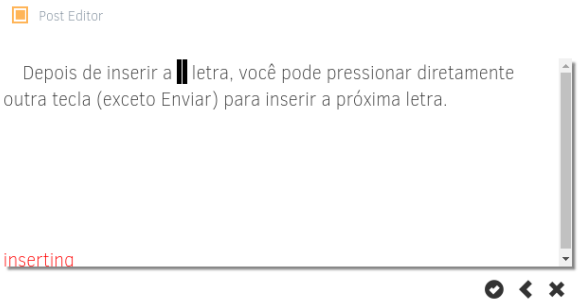
(b) Typeahead component screenshot two.

Figure 3.12: Screenshots of the dictionary-backed typeahead component used as the contrastive setting in the post-editing actions user study

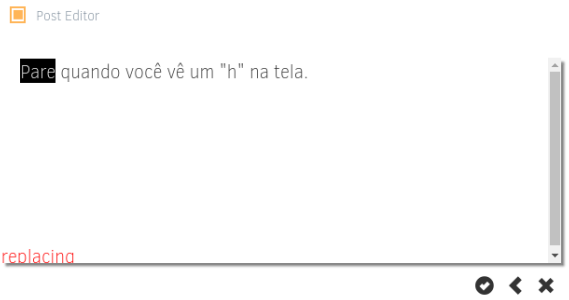
The typeahead and post-editor components were implemented as *target-area* components, conforming to the HandyCAT target area interface discussed in the first part of this chapter. Figures 3.12 and 3.13 present screenshots of the final component implementations after several design iterations.

The baseline typeahead component described in section 3.6 was used as the contrastive setting in the experiments described below. The auto-completion service was seeded with the top 20,000 words from a large parallel Portuguese–English corpus, collected from the OPUS corpus (Tiedemann, 2012). When a user begins typing in the typeahead text area, the top five suggestions from this corpus are shown. There is no language-modeling or context-aware aspect to the auto-complete model, suggestions are simply sorted by (1) their overlap with the user's current prefix, and (2) their frequency in the corpus.

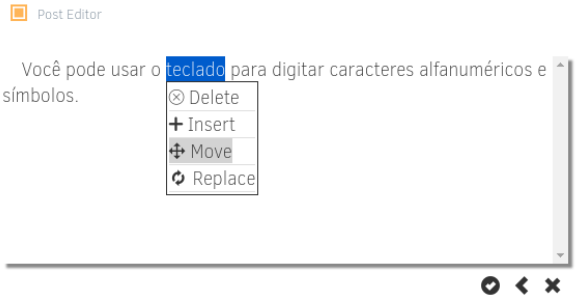
The post-editing component implements each of the editing modes described above. The mouse is utilized to select text and to choose the desired editing modes. The component recognizes word boundaries, so simply clicking inside a word selects the entire word for editing. Users can also click on whitespace to insert text, or remove unnecessary



(a) Post-editor component INSERT mode.



(b) Post-editor component REPLACE mode.



(c) Post-editor component MOVE mode.

Figure 3.13: Screenshots of the INSERT, REPLACE, and MOVE modes in the post-editing component

whitespace.

The INSERT, REPLACE, and MOVE modes are two-step processes; in other words, the user selects the mode, then performs an action while inside that mode. In the INSERT and REPLACE modes, the user must explicitly exit by pressing the Escape key, the MOVE mode is automatically exited when the user chooses the location where the selected text should be moved. In contrast, the DELETE action is a one-step operation – the interface immediately returns to the default state once the user's selected text has been deleted.

Experimental Design

Approximately 50 native Portuguese speakers with experience translating from English to Portuguese participated in the post-editing user study. These users were distributed across three groups, three texts, and the two edit modes autocomplete (AC) and post-edit (PE). Table 3.9 shows the distribution of users by session, text, and editing mode. The user logs were obtained from three groups in sessions conducted by Felix do Carmo at the University of Porto.

Group	# Participants	Session	Project	Mode
1	17	A	Text B – Questionnaire	AC
	17	B	Text C – Catalogue	PE
	17	C	Text D – Manual	AC
	18	D	Text D – Manual	PE
2	16	A	Text C – Catalogue	AC
	17	B	Text D – Manual	PE
	18	C	Text B – Questionnaire	AC
	17	D	Text B – Questionnaire	PE
3	10	A	Text D – Manual	AC
	10	B	Text B – Questionnaire	PE
	13	C	Text C – Catalogue	AC
	13	D	Text C – Catalogue	PE

Table 3.9: Distribution of users by session, project, and mode (data from do Carmo (2017))

In order to obtain log information about the users' interactions with the post-editor component, several component-specific actions were added to the HandyCAT log format.

3.7.1 Results

This section describes the results of the user study. Data was obtained from the results in do Carmo (2017) where indicated – a more thorough analysis of the results is also provided in do Carmo (2017).

Project	AC	PE	Average
Questionnaire	11	15	12
Catalogue	9	17	13
Manual	10	16	14
Average	10	16	13

Table 3.10: Average editing speed (in seconds) per segment for each project and mode

Users were faster in AC mode than they were in the PE mode (table 3.10). This is not surprising, since our implementation of PE mode required frequent use of the mouse. The PE mode also introduced a novel interface that is not similar to any existing tools, so users likely required some time to learn this unfamiliar way of editing (in spite of the introductory training session).

Action	Session B	Session D
delete	9	6
insert	14	13
move	14	9
replace	16	11

Table 3.11: Average duration by edit action type (i.e. the average time spent in a particular edit mode) in session B vs session D, in seconds.

Table 3.11 shows the average duration for each of the editing actions. This is the average time that a user took to perform one of the actions. Again, as expected, the multi-step editing actions INSERT, MOVE, and REPLACE tended to take significantly more time than DELETE.

Action	# Events
delete	1580
insert	810
move	425
replace	1620

Table 3.12: Total number of events for each of the editing actions

Table 3.12 shows the raw count for each of the edit action types. The action counts reveal that DELETE and REPLACE were heavily favored over INSERT and MOVE. We hypothesize that REPLACE is preferred over INSERT because it is more common for users to make *modifications* than it is for them to make *additions*. The MOVE action is arguably the most complex, and is also the least used of the four.

The amount of editing done by each user, as measured by TER was fairly consistent regardless of the editing mode used, implying that translations are not heavily biased by the editing component. Again, we take this as a positive result supporting macro operations designed for post-editing.

3.7.2 Key Takeaways from PE Actions User Study

This user study provided a basic understanding of translators' preferences for different editing operations when post-editing, and produced a large amount of data logging translators' interactions with the post-editing component. Useful feedback on the interface itself was also obtained through the post-surveys, with generally positive responses from most participants.

The development work done during the implementation of the post-editing component also served as the basis for the interactive post-editing study discussed in chapter 7, which evaluates an *interactive* post-editing interface, where translators collaborate with a specialized machine translation system, and word-level quality estimation models while post editing.

3.8 Discussion

This chapter has described HandyCAT: a purpose-built web-based interface for translation process research. The reasoning behind our decision to implement a CAT tool from scratch was put forward in section 3.1.2, and a novel design methodology, called component-centric design (CCD), was proposed and formally described in section 3.2.1. CCD in particular motivated the implementation of HandyCAT, and the flexibility and robustness of the framework has been demonstrated by the diverse components that have been implemented and tested in HandyCAT.

We have shown how HandyCAT implements the CCD framework, and discussed the

concept of component areas, which allows for very fast prototyping and integration of new user interface components into the tool.

The implementation of a modern web-based CAT tool from scratch was quite an ambitious undertaking, and many lessons were learned along the way. HandyCAT underwent three major rewrites, as well as many small modifications for each user study that was conducted with the interface. The tool itself, and all of the supporting microservices for each of the user studies we conducted are open-source[14].

The final sections of this chapter introduced two pilot user studies conducted with HandyCAT. Building upon the lessons learned, and ideas gleaned from these user studies, the chapter 7 of this thesis presents a HandyCAT implementation of an advanced component for interactive post-editing, and an accompanying user study evaluating this component.

[14] `https://github.com/chrishokamp/handycat`

Chapter 4

Predicting Translation Quality at the Word-Level

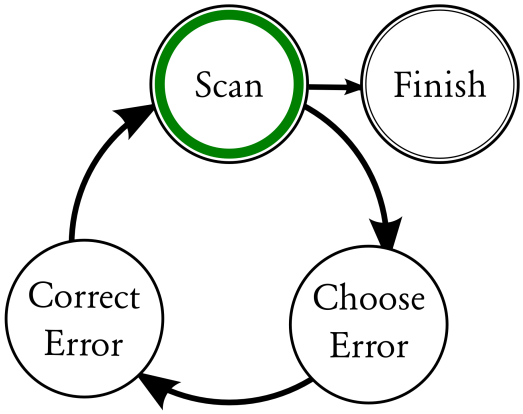


Figure 4.1: The Post-Editing Cycle

4.1 Introduction

In this chapter, we explore models which can predict translation quality. Specifically, we are interested in models which can annotate translations with labels or scores at the token or word level. Focusing upon *RQ2*, we try to design new models which can achieve state-of-the-art results in word-level QE tasks. As discussed in section 2.5, in this class of tasks, we are given a source segment and a corresponding translation hypothesis, and we wish to output a sequence of labels, where each label corresponds to one token in the machine translation hypothesis. In other words, the word-level QE task is an instantiation of a *sequence labeling* task (Lafferty et al., 2001; Nguyen and Guo, 2007). Each label

may additionally include a value between zero and one, indicating the model's confidence about its prediction.

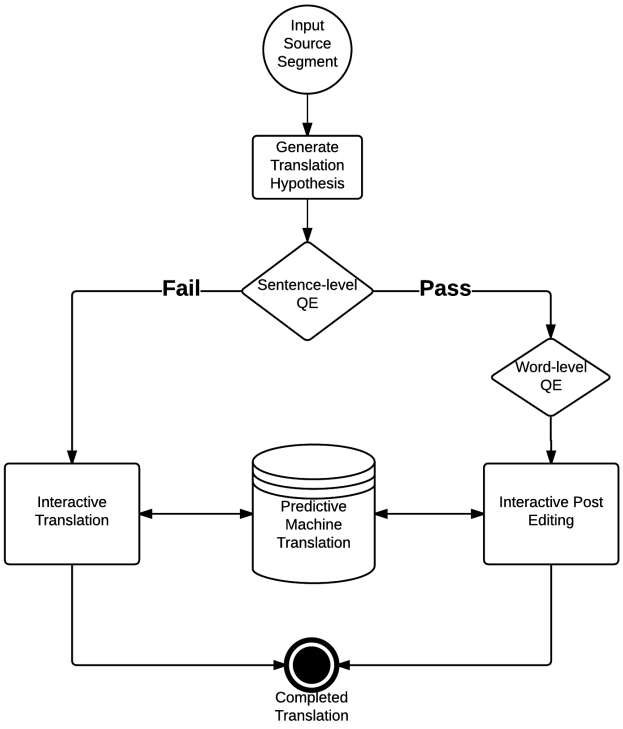


Figure 4.2: One possible interactive workflow for CAT, supported by both word- and sentence-level quality estimation.

The most obvious real-world use case for such models is as a guide for translators when performing post-editing. *Scanning*, or searching for an error, is the first step in the post-editing cycle (figure 4.1) – to streamline this step we seek a high-quality model which can point to problem spots in the translation output, guiding translators towards the places which need to be changed. Therefore, QE can help during the *Scan* and *Choose Error* phases of the PE cycle depicted in figure 4.1.

Recent work has also shown that accurate word level models of translation quality are good proxies to sentence-level models — this result is in line with our intuition that a translation with many individual errors should also have low overall quality (Bojar et al., 2017; Martins et al., 2017). However, despite the direct connection between word-level and sentence-level quality, the use of word-level QE models for sentence-level QE may discard information about the contextual *severity* of individual errors, which could potentially be important when making accurate sentence-level quality judgments.

Another important real-world use case for QE models in general is as "gatekeepers" in

translation workflows. In this scenario, a QE model can be used to determine whether or not a translation hypothesis needs to be reviewed by a human post-editor. An accurate quality estimation model used in conjunction with a machine translation system of adequate quality can potentially streamline translation projects by allowing translations with no detected quality issues to bypass the post-editing phase completely. Figure 4.2 is a schematic of a translation process workflow which employs sentence level QE as a sentinel, and word-level QE as an interactive aid during post-editing.

As discussed in section 2.5, our work focuses upon the binary labeling of each word in a translation system's output as OK or BAD. However, as discussed in section 2.5, other tagging schemes, such as hierarchical tagsets using MQM issues (Lommel et al., 2014), are possible. However, because the majority of existing work on word level QE focuses upon the binary classification case, and because training and test data for the task can be automatically created, all of our models and experiments focus upon this version of the task.

The rest of this chapter is organized as follows: section 4.1.1 reviews the methods used to create datasets for QE models. Section 4.1.2 discusses some of the standard features used for training linear QE models as classifiers or structured predictors, and introduces our open-source project *Marmot*[1], which is a Python library designed to streamline complex feature extraction workflows. Section 4.2.1 gives an overview of the formerly state-of-the-art models which use feature extraction coupled with structured predictors for sequence labeling, while section 4.2.2 presents our research into word-level QE models which rely upon deep learning. Section 4.3 delves into a unified view of Automatic Post-Editing and Word-Level QE, which has recently achieved state-of-the-art results in both the APE and QE tasks. Finally, section 4.4 summarizes the work conducted on QE and related tasks, and highlights the main contributions made by our research. Much of the research presented in this chapter was conducted in collaboration with Unbabel, and with researchers at the University of Sheffield. Therefore, when a topic or result is discussed that was joint work, or not our contribution, we cite the relevant publications explicitly.

[1] `https://github.com/qe-team/marmot`

SRC	A class stores information about the types of data that an object can hold and the behaviors that an object can exhibit .
MT	Eine Klasse speichert Informationen über die Datentypen , die ein Objekt , und halten Sie die Verhaltensweisen , die ein Objekt erzeugen können .
PE (Reference)	Eine Klasse speichert Informationen über die Datentypen , die ein Objekt aufnehmen kann , und die Verhaltensweisen , die ein Objekt zeigen kann .
Gold Tags	OK OK OK OK OK OK OK OK OK OK OK BAD BAD BAD BAD OK OK OK OK OK OK BAD BAD OK

Table 4.1: Source, MT hypothesis, Post-edited reference, and gold-standard binary QE labels obtained using TER. The example is taken from line 38 of the WMT 2016 QE test data.

4.1.1 Building Datasets for Word-Level QE

In order to train QE Models in a supervised fashion, we require datasets of triples: $(\mathrm{SRC}, \mathrm{MT}, \mathrm{PE})$, where:

- SRC: the original source sequence to be translated
- MT: a machine translation hypothesis produced by inputting the SRC into an automatic translation system
- PE: a post-edited version of MT, typically produced by a human translator with access to both SRC and MT

The post-edited sequences can be viewed as a special kind of *constrained* reference, which is as close as possible to the original MT hypothesis, while still being a human-quality reference translation. This observation stems from the hypothesis that post-editors will seek to minimize the effort required to produce the final translation, and thus will generally try to make the minimum number of required edits. This constraint that references and hypotheses should have the minimum edit distance will also be used to motivate the intuition behind synthetic data generation for QE and APE discussed later in the chapter.

4.1.2 Marmot: A Framework for Word-Level Quality Estimation

The input to many word level QE models is a sparse vector with thousands or millions of features. These state-of-the-art feature sets require complex feature extraction pipelines. Some features, such as those extracted from constituency or dependency parse structures, may be time-consuming to compute, while others, such as part-of-speech n-grams, may

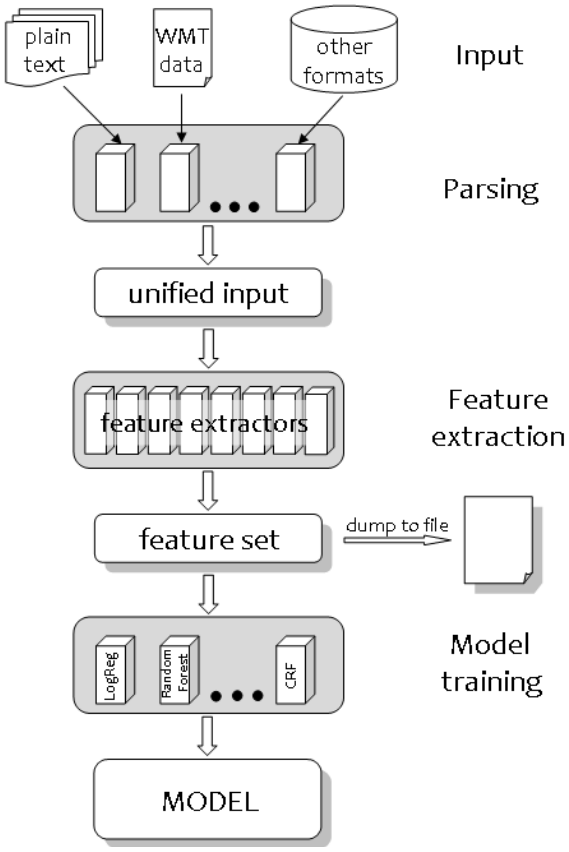


Figure 4.3: The data extraction pipeline used in MARMOT (from Logacheva et al. (2016))

require a pipeline of extractors to run in a specific order (i.e. part-speech-tagging must run before n-gram extraction). Some groups of features may have no dependencies, and can thus be computed offline, or in parallel. Because configuration and implementation of feature extraction pipelines is time-consuming and error-prone, several open-source toolkits have been built to simplify and standardize the feature extraction process specifically for QE.

The QuEST framework (Specia et al., 2013b; Shah et al., 2013; Specia et al., 2015) was the first open-source toolkit targeted at QE. The framework provides a Java-based pipeline for feature extraction, as well as a python library for training machine learning models using Scikit-learn (Pedregosa et al., 2011). QuEST is targeted at sentence-level QE, but an extended version, called QuEST++, also supports extraction of word-level and document-level features (Specia et al., 2015). In order to run feature extraction in QuEST, the user specifies which features are desired using an XML configuration file. The features in the baseline set used in the WMT QE tasks can all be extracted by QuEST – for the 2014, 2015, 2016, and 2017 WMT QE tasks, the QuEST framework was used

Feature Index	Description
1	number of tokens in the source sentence
2	number of tokens in the target sentence
3	average source token length
4	LM probability of source sentence
5	LM probability of target sentence
6	number of occurrences of the target word within the target hypothesis (averaged for all words in the hypothesis - type/token ratio)
7	average number of translations per source word in the sentence (as given by IBM 1 table thresholded such that prob(t|s) > 0.2)
8	average number of translations per source word in the sentence (as given by IBM 1 table thresholded such that prob(t|s) > 0.01) weighted by the inverse frequency of each word in the source corpus
9	percentage of unigrams in quartile 1 of frequency (lower frequency words) in a corpus of the source language (SMT training corpus)
10	percentage of unigrams in quartile 4 of frequency (higher frequency words) in a corpus of the source language
11	percentage of bigrams in quartile 1 of frequency of source words in a corpus of the source language
12	percentage of bigrams in quartile 4 of frequency of source words in a corpus of the source language
13	percentage of trigrams in quartile 1 of frequency of source words in a corpus of the source language
14	percentage of trigrams in quartile 4 of frequency of source words in a corpus of the source language
15	percentage of unigrams in the source sentence seen in a corpus (SMT training corpus)
16	number of punctuation marks in the source sentence
17	number of punctuation marks in the target sentence

Table 4.2: The 17 baseline features used in the WMT Sentence Level QE tasks

to extract the baseline features distributed to participants in the sentence level task (an overview of these features is given in table 4.2).

Noting the lack of a fully-featured QuEST equivalent for word-level QE, we built *Marmot* [2] (Logacheva et al., 2016) as a python-based alternative to QuEST. Our contribution to this framework included the design and implementation of the domain-specific language (DSL) that enables pipeline configuration, as well as parts of the model training pipeline for deep models and structured predictors. We also implemented several of the feature extractors, and wrote much of the online documentation. Like QuEST, Marmot supports feature extraction, model training, and evaluation, but is targeted primarily at tasks which seek to output translation quality labels at the word level. Figure 4.3 is a high-level sketch of the entire pipeline enabled by Marmot.

Marmot is targeted towards Python programmers, providing very flexible configuration interfaces, and a set of well-defined interfaces for different types of feature extractors. Marmot's built-in extractors are easily extendable – configuration is done with a Domain

[2] https://github.com/qe-team/marmot

Feature Index	Description	Type
1	number of tokens in the source	Global
2	number of tokens in the target	Global
3	ratio of source and target sentences length	Global
4	target word	Local
5	left context of the target word	Local
6	right context of the target word	Local
7	source word aligned to the target word	Local
8	left context of the aligned source word	Local
9	right context of the aligned source word	Local
10	target word is stopword	Local
11	target word is punctuation mark	Local
12	target word is proper noun	Local
13	target word is digit	Local
14	highest order of ngram that includes target word and its left context	Local
15	highest order of ngram that includes target word and its right context	Local
16	backoff behavior of ngram $w_{i-2}w_{i-1}w_i$ (w_i is target word)	Local
17	backoff behavior of ngram $w_{i-1}w_iw_{i+1}$ (w_i is target word)	Local
18	backoff behavior of ngram $w_iw_{i+1}w_{i+2}$ (w_i is target word)	Local
19	highest order of ngram that includes source word and its left context	Local
20	highest order of ngram that includes source word and its right context	Local
21	POS of the target word	Local
22	POS of the aligned source word	Local
23	target word + left context	Local
24	target word + right context	Local
25	target word + aligned source word	Local
26	POS of target word + POS of aligned source word	Local
27	target word + left context + source word	Local
28	target word + right context + source word	Local

Table 4.3: The baseline features used in the WMT Word Level QE tasks. In the **Type** column, features with type **Global** are the same across every token in a sentence, whereas **Local** features are extracted for each word individually

Specific Language (DSL) which specifies a topology of feature extractors that will process the raw data. Listing 4.1 shows an example of the Marmot's DSL, which is an extension of YAML[3]. A key aspect of feature extraction pipelines in general is that some extractors depend on certain data being available. For example, the feature POS-Left (i.e. the part-of-speech of the word to the left of the current word) depends upon the POS tags for the sequence being available. Marmot handles such dependencies elegantly, by allowing the user to specify which feature extractors must run before a particular feature can be computed, and by failing cleanly and informatively when a feature-extractor's dependencies are not met at run time.

[3]http://yaml.org/

Listing 4.1: An example of Marmot's configuration DSL

```
---
context_creators:
# a positive corpus context creator
  - module: marmot.util.corpus_context_creator.CorpusContextCreator
    args:
      - type: function_output
        func: marmot.preprocessing.parsers.parse_wmt14_data
        args:
          - !join ../examples/word_level_quality_estimation/data/en_es/EN_ES.tgt_ann.train
          - !join ../examples/word_level_quality_estimation/data/en_es/EN_ES.source.train
          - type: function_output
            func: marmot.preprocessing.parsers.extract_important_tokens
            args:
              - !join ../examples/word_level_quality_estimation/data/en_es/EN_ES.tgt_ann.test
              - 1

# require contexts to satisfy these filter constraints in order to be included in classifier training
filters:
  min_total: 2
  min_class_count: 1

# feature extractors are used to map over contexts
feature_extractors:
  - module: marmot.features.token_count_feature_extractor.TokenCountFeatureExtractor
  - module: marmot.features.alignment_feature_extractor.AlignmentFeatureExtractor
    args:
      - ''                                          # alignment model
      - !join ../examples/word_level_quality_estimation/data/en_es/europarl.1000.en
# parallel corpus - source
      - !join ../examples/word_level_quality_estimation/data/en_es/europarl.1000.es
# parallel corpus - target
  - module: marmot.features.dictionary_feature_extractor.DictionaryFeatureExtractor
    args:
      - 'spanish'              # target language for stopwords extraction
  - module: marmot.features.lm_feature_extractor.LMFeatureExtractor
    args:
      - !join ../examples/word_level_quality_estimation/data/en_es/europarl.1000.es # file for LM
      - 3                                           # LM order
  - module: marmot.features.pos_feature_extractor.POSFeatureExtractor
    args:
      - !join ../examples/word_level_quality_estimation/data/en_es/tree-tagger
      - !join ../examples/word_level_quality_estimation/data/en_es/english-utf8.par
      - !join ../examples/word_level_quality_estimation/data/en_es/spanish-par-linux-3.2-utf8.bin

...
---
```

Marmot represents each data instance as a *context object*, which is a Python `dict` that can be serialized into a JSON object containing all of the extracted features and metadata about the training instance. This allows training instances, which begin as raw text, to

System ID	F_1-**Bad**
UAlacant/OnLine-SBI-Baseline	43.12
HDCL/QUETCHPLUS	43.05
UAlacant/OnLine-SBI	41.51
Baseline features + VW classifier	40.84
SAU/KERC-CRF	39.11
SAU/KERC-SLG-CRF	38.91
• SHEF2/W2V-BI-2000	38.43
• SHEF2/W2V-BI-2000-SIM	38.40
SHEF1/QuEst++-AROW	38.36
UGENT/SCATE-HYBRID	36.72
• **DCU-SHEFF/BASE-NGRAM-2000**	36.60
HDCL/QUETCH	35.27
• **DCU-SHEFF/BASE-NGRAM-5000**	34.53
SHEF1/QuEst++-PA	34.30
UGENT/SCATE-MBL	30.56
RTM-DCU/s5-RTM-GLMd	23.91
RTM-DCU/s4-RTM-GLMd	22.69
• **Baseline features + CRFSuite**	16.78

Table 4.4: Official results for the WMT15 Quality Estimation Task 2. Systems whose results are significantly different with $p = 0.05$ are grouped by a horizontal line. Systems **in bold** used the baseline feature set. Systems that used Marmot toolkit for feature extraction and/or model training are indicated by a •.

System ID	F_1-**Bad**
UAlacant/OnLine-SBI	41.51
UAlacant/OnLine-SBI + Baseline	**43.12**
HDCL/QUETCH	35.27
HDCL/QUETCH + Baseline	**43.05**

Table 4.5: Performance of WMT15 systems with and without the baseline features.

easily pass through a pipeline of feature extractors, where each successive extractor adds metadata or mutates fields on the instance. Once feature extraction is finished, the extracted features can be written to a file in a representation suitable for input into machine learning models, such as tab-separated value (TSV), or CoNLL (Buchholz and Marsi, 2006) format.

Marmot also provides a programmatic interface to a variety of machine learning models, and can dump features in formats which support several libraries for structured prediction, including the CRFSuite[4] toolkit, which has been used as the baseline model for

[4] `http://www.chokkan.org/software/crfsuite/`

the last three iterations of the WMT word level QE task. Table 4.4 shows the results of the WMT 15 QE tasks, with systems that used Marmot marked with a bullet point. Table 4.5 shows the impact of the baseline feature set extracted using Marmot on the top two systems. For a more detailed discussion see Logacheva et al. (2016).

4.2 Investigating Model Types for Word-Level QE

The following sections present a series of experiments investigating various models and features for the word-level QE task.[5]

4.2.1 Feature-Based Models for Word Level QE

Martins et al. (2016, 2017) use a linear sequence model trained using the max-loss MIRA algorithm (Crammer et al., 2006). This model includes first-order edge features, which are paired labels that are possible transitions between the output tags (y_i, y_{i-1}), in addition to the baseline features in Table 4.3. Following Kreutzer et al. (2015b), we expand the features available to the baseline model by pairing target words with each observed left and right context, and aligned source word, and pairing each target POS tag with its aligned source POS tags. The resulting system provided a robust baseline which outperformed 15 of the 17 systems submitted to WMT 15 (see Table 4.7 below).

An advantage of linear sequence models is the ease of adding new features, each of which may come from another standalone model. This ensembling approach is the current state of the art in word level QE (Martins et al., 2017); however, it has the disadvantage of adding significant complexity to the model training and inference pipelines. Therefore, we were interested to find single models that can be trained end-to-end, ideally with minimal pre-processing or feature extraction. This is the motivation behind the deep learning based models discussed in the following sections.

4.2.2 Deep Models for Word-Level Quality Estimation

Seeking both to improve word-level QE, and to simplify the training and inference steps of feature-based models, we wish to find a machine learning model which can take ad-

[5]Much of the work described in this section was joint work led by the Unbabel research team, conducted during two secondments in 2015 and 2016. Martins et al. (2016) and Martins et al. (2017) are our joint publications with this group.

vantage of the rich information available in very large parallel texts, while also modeling dependencies within and between sequences in the source and target languages.

Recurrent Neural Networks (Sutskever et al., 2014a) are a good initial choice because they have been shown to model complex dependencies in sequential inputs. We cannot directly use a recurrent neural network for the word level QE task, because our model is a function $(\mathrm{SRC}, \mathrm{HYP}) \rightarrow \mathrm{TAGS}$, where $\mathrm{SRC} = s_0 \dots s_{|\mathrm{SRC}|}$, $\mathrm{TRG} = t_0 \dots t_{|\mathrm{TRG}|}$, and $\mathrm{TAGS} = \hat{y}_0 \dots \hat{y}_{|\mathrm{TRG}|}$. In other words, we wish to output a tag for each token in the target sequence, but we also need to take the source sequence into account.

Therefore, we begin by considering a monolingual, bi-directional recurrent model with access only to the target sequence as the inspiration (and the baseline) for the models described in this section. Our baseline bidirectional recurrent model (Schuster and Paliwal, 1997) is an extension of the recurrent language model (Bengio et al., 2003; Mikolov et al., 2011) which concatenates the representations of a left-to-right recurrent model with the right-to-left representation (Bahdanau et al., 2014). The hidden states of this model can be used to represent each word in the target hypothesis for word-level QE because the entire sequence is available to the model as input, in contrast with many other applications of language modeling in the context of sequence to sequence models, where the target sequence generated from beginning to end, thus precluding the possibility of a bidirectional model for the output sequence.

During training, we wish to minimize the cross-entropy loss of the predictions $\hat{\mathbf{y}}$ compared with the gold-standard tags $\mathbf{y}$, e.g. equation 4.1:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{T} \mathbf{y}_{ij} \log \mathbf{p}_{ij}, \tag{4.1}$$

where each $\mathbf{p}_{ij}$ is a vector containing the model's distribution over the tags $\mathrm{OK}, \mathrm{BAD}$ for timestep j of input i. Because our task is binary, we have taken a notational shortcut in Eq. 4.1, and implied that $\mathbf{y}_{ij} = 1$ if the tag is OK, and that $\mathbf{y}_{ij} = 0$ if the tag is BAD. Note also that, because the classification task is binary, we could have used the binary cross-entropy loss instead of the log loss, but the models were implemented with log loss in order to keep the implementation as general as possible, to support an arbitrary number of output tags if needed.

4.2.3 Adding Source Alignment Information

In order to take advantage of the source input, while still using only a single input sequence, the aligned source word for each target word is included as an additional input, with its own embedding, similar to the approach taken by QUETCH (see section 2.5.4). The input to the model at each timestep thus becomes a tuple with two items: the target word and its aligned source word. Our model now has access to both the source language context and the target language context. Because of its the similarity with recurrent language models (Mikolov et al., 2011), we call this model the *bilingual, bidirectional language model* (BBLM).

We validate that adding source information improves performance considerably by evaluating a baseline model with access only to the target sequence (see table 4.6). By simply including the first aligned source token for each target word into the recurrent representation, we were able to outperform the majority of the submissions to the WMT 15 QE task, without using any other features. As discussed above, most of these models use a large suite of features, which require complex pre-processing pipelines to compute.

Note that the addition of alignment information assumes that the word alignments of the MT system are available. In other words, the inclusion of this information treats the MT system as a "translucent box", which provides target-to-source word alignments as outputs in addition to translation hypotheses. Although the WMT task organizers do provide this information, we cannot always assume that it will be available. The following section addresses this issue head-on, with a specific solution proposed in Section 4.3.1.

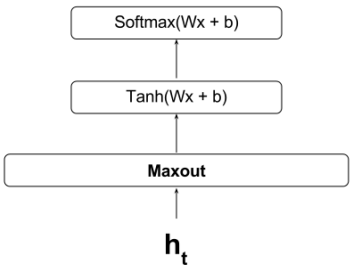


Figure 4.4: Schematic of an output layer which includes a maxout transformation.

The best-performing architecture (figure 4.5) is a BBLM which uses a multilayer perceptron with a maxout (Goodfellow et al., 2013) transformation at the first layer to output classification decisions (figure 4.4). When used together with dropout (Srivastava et al.,

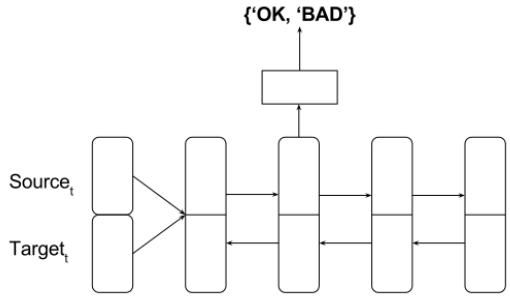


Figure 4.5: Schematic of the Bidirectional Bilingual Language Model

System	Dev			Test		
	$F1_{bad}$	$F1_{good}$	$F1_{weighted}$	$F1_{BAD}$	$F1_{GOOD}$	$F1_{weighted}$
WMT Baseline	-	-	-	.168	.889	.753
BLM	.384	.644	.594	.373	.648	.596
BBLM	.412	.740	.678	.403	.747	.682
BBLM+Maxout	**.429**	.738	.682	**.417**	.743	.644

Table 4.6: BLM (baseline) is a bidirectional recurrent language model using only target-language information. BBLM includes aligned source words in in the input to the bidirectional model. BBLM+Maxout includes a Maxout layer followed by two linear layers for a deeper output layer of the model. Bold fonts are significant improvement over the BLM baseline.

2014), an MLP which starts with a maxout transformation performs better than a simple linear transformation with bias. This model outperforms all but two of the submissions to WMT 2015 with only the target sequence and the alignment information.

4.2.4 Experiments

For comparison purposes, the QUETCH implementation in (Kreutzer et al., 2015b) was reproduced in Martins et al. (2016). The same training setup as the original model was followed. The best model used the pre-trained word embeddings released by the Polyglot project (Al-Rfou et al., 2013) and a hidden layer of size 10. Table 4.8 shows the results of the QUETCH re-implementation.

Recurrent model configuration

All recurrent models use the Gated Recurrent Unit (GRU) for the recurrent transitions (Chung et al., 2015). The size of the hidden layers is fixed at 500, and embedding size is set to 200. Embeddings are initialized with embeddings from an English–Spanish neural machine translation system trained with our own NMT framework based on Blocks (van

System	Dev			Test		
	$F1_{bad}$	$F1_{good}$	$F1_{weighted}$	$F1_{BAD}$	$F1_{GOOD}$	$F1_{weighted}$
Linear+Seq	.417	.794	.721	.416	.780	.727
Linear+Seq+Q*	.427	.795	.725	.425	.800	.730
Linear+Seq+BBLM	.431	.793	.724	.423	.799	.728
Linear+Seq+BBLM+Q	**.439**	.793	.726	**.432**	.800	.731

Table 4.7: Results of Linear Sequence Model and Ensemble Models: Q=QUETCH feed-forward model, BBLM=Bidirectional LM with source alignments. * indicates our re-implementation of the best performing system from WMT 15

F1-BAD Scores	Dev	Test
QUETCH	*	.387
re-implemented	.414	.416

Table 4.8: Reported QUETCH scores vs. the implementation in Martins et al. (2016). The discrepancy in scores may be due to differences in training configuration, or the pre-trained word embeddings used to initialize the model.

Merriënboer et al., 2015)[6], which achieves 29.74 BLEU on the WMT 2013 `newstest` dataset. We experimented with tuning the embedding parameters during training, but found that leaving embedding parameters static improves performance.

All hyper-parameters were optimized using the development set provided by task organizers. Following Kreutzer et al. (2015a), we weight the loss for BAD instances by a factor of 5.0, effectively increasing the penalty of the model when BAD instances are mis-classified more than the penalty for OK instances.

For the recurrent models, mini-batch size is fixed at 40. Dropout is applied to all feed-forward parameters of the models. We test the impact of L2 regularization, and our best performing system uses both dropout and L2 regularization[7]. All recurrent models are optimized using AdaDelta (Zeiler, 2012b).

Our ensemble model uses the same configuration as the baseline system, with one or more features included from the deep models (Table 4.7).

All experiments were conducted on the WMT 15 English–Spanish word-level QE dataset. Following the WMT 2015 evaluation criteria, system ranking is determined by F1 score for the BAD class on the WMT15 English–Spanish test data. For all experiments, the final models were chosen using the official WMT 2015 development data.

[6]Our implementation is available at `https://github.com/chrishokamp/neural_mt`

[7]peak performance was obtained with dropout probability set to 0.5, and L2 regularization $\alpha = 10^{-4}$

4.2.5 Recurrent Models for Word-Level QE – Discussion

Table 4.6 shows results for our recurrent model configurations, and table 4.7 shows the results for ensemble models.

Our best ensemble model significantly outperformed the existing state of the art, but not by a very large margin. Because the recurrent models in particular have a very large capacity, and because our ensemble models are already using millions of features with significant tuning to achieve relatively small performance gains, we believe that all of the top performing models are close to the limit of what can be learned from the task-internal training data.

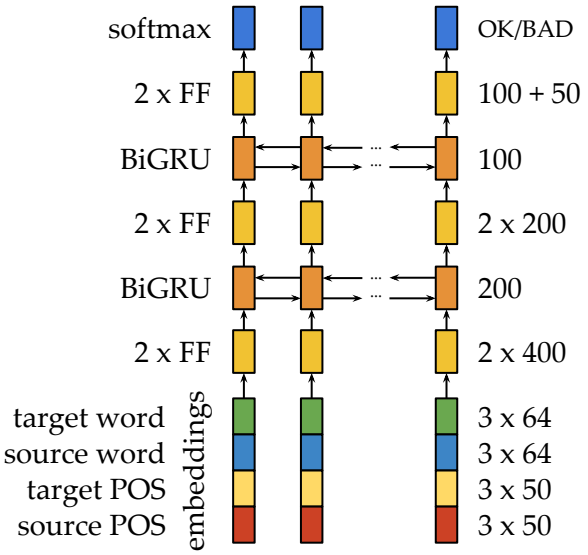


Figure 4.6: The best performing bidirectional recurrent model from Martins et al. (2017)

The preceding set of experiments evaluated a suite of models for the word-level QE task, and introduced a novel bidirectional, bilingual langugage model (BBLM) which significantly outperformed existing deep models. When BBLM output is included into an ensemble with the baseline features, our models achieved state-of-the-art performance on the WMT 2015 word-level QE dataset, when compared to other single-model architectures. We also showed that source and target word embeddings trained by a neural machine translation system can be used to improve model performance. Follow-on work to this project indicated that deeper recurrent models with more features included as factors can further improve performance. Figure 4.6 depicts the best recurrent architecture found by Martins et al. (2017), which is a component of an ensemble that achieved a new

state-of-the-art result.

4.3 Unifying Word Level Quality Estimation and Automatic Post-Editing

Chapter 2 briefly pointed out that automatic post-editing is closely related to word-level QE. The input to APE and QE systems is identical: a source sequence with a corresponding target hypothesis. However, instead of labeling the words in the hypothesis as OK or BAD, APE systems are expected to directly output a revised translation hypothesis, which is the APE model's best guess about what the translation hypothesis will look like *after* it has been post-edited. QE systems, on the other hand, only need to output labels for each token of the MT hypothesis, they are not required to predict *how* any BAD labels should be fixed.[8] Thus, when QE labels correspond to the edit operations *insertion* or *substitution*, it is trivial to use an APE system for QE, since labels can be obtained by computing an edit-distance alignment such as TER between the MT hypothesis and the APE system output.

Recent work (Junczys-Dowmunt et al., 2016; Martins et al., 2017) has confirmed that good APE models are also good word level QE models. By computing the edit distance between an APE system's hypothesis and the original MT hypothesis, we can obtain labels for each hypothesis token. This method has the additional benefit that it corresponds exactly with the methodology for creating the training and gold-standard data for the word level QE task in standard QE datasets such as those used in the WMT competitions, except that gold-standard data is created from human post-edits, instead of pseudo-references.

As discussed in the first part of this chapter, state-of-the-art systems which use sequence labeling techniques to perform word-level quality estimation are currently reliant upon system-internal word alignment information. Although the training data used for the MT systems in the WMT QE tasks is not available, the word alignments are part of the released data, and state-of-the-art methods critically rely upon this information. The top performing model in WMT 16 uses the alignment between MT tokens and source tokens to create the input representation for both recurrent and the feed-forward models. Without

[8] QE systems in their current formulation also cannot account for *deletions* in the hypothesis, because they only label the tokens that are present

the word alignment information, this input cannot be created.

We are interested in removing this dependence upon system-internal information, and in leveraging state-of-the-art approaches to both APE and QE — we hypothesize that features which have been shown to be beneficial for quality estimation are also likely to benefit APE models. In order to test this hypothesis, the following sections propose including QE features such as *predicted* word alignments, POS tags, and dependency relations, as *factors* in the input of a neural APE system. We also propose ensembling one or more systems with factored inputs together with SRC-PE and MT-PE decoders to create a multi-model decoder with diverse input representations.[9]

As discussed in sections 2.5 and 2.6, two important lines of research have recently made breakthroughs in APE and QE, which can be summarized as (1) techniques for augmenting APE training datasets with synthetic data, and (2) multi-model ensembles for word-level QE, which use large, sparse feature sets. Below we briefly review the key aspects of these breakthroughs, because our contribution builds upon them.

Data Augmentation for APE

As we have understood, APE and QE training datasets consist of (source, target, reference) triples, where the post-edited reference is created by a human translator. However, publicly available APE datasets are relatively small in comparison to parallel datasets used to train machine translation systems. Junczys-Dowmunt and Grundkiewicz (2016) introduce a method for generating a large synthetic training dataset from a parallel corpus by first translating the reference to the source language, and then translating this "pseudo-source" back into the target language, resulting in a "pseudo-hypothesis" which is likely to be more similar to the reference than a direct translation from source to target. The MT component of the synthetic triple thus comes from this round-trip translation process, while the source and post-edit components correspond to the source and reference from the original parallel corpus. The process of generating the "pseudo-hypothesis" is depicted in figure 4.7. The release of a large synthetic training dataset by Junczys-Dowmunt et al. (2016) was a major contribution towards improving APE.

[9]code available at `https://github.com/chrishokamp/constrained_decoding` and `https://github.com/chrishokamp/qe_sequence_labeling`

Figure 4.7: The flow of creating synthetic data for APE, proposed by Junczys-Dowmunt et al. (2016). The reference side of a parallel corpus is translated to the source language, using an MT system from $\mathbf{TRG} \rightarrow \mathbf{SRC}$. This "pseudo-source" is then translated back into the target language using a $\mathbf{SRC} \rightarrow \mathbf{TRG}$ MT system. The resulting "pseudo-hypothesis" should be close to the reference, but is likely to have noise introduced by passing through two MT systems.

Junczys-Dowmunt and Grundkiewicz (2016) also present a framework for ensembling SRC $\rightarrow$ PE and MT $\rightarrow$ PE NMT models together, and tuning for APE performance, an idea which we extend in the following sections.

Stacked Models for word-level QE

Martins et al. (2016) introduced a stacked architecture, which uses features generated by several sub-models based on neural networks, as well as a very large feature set including many of those in table 4.3 as well as dependency parse features, within a structured prediction framework, achieving a large jump in the state of the art for word-level QE. Some features are actually the outputs of standalone feed-forward and recurrent neural network models, which are then stacked into the final system. Although their approach creates a very good final model, the training and feature extraction steps are quite complicated. An additional disadvantage of this approach is that it requires "jackknifing" the training data for the standalone models that provide features to the stacked model, in order to avoid over-fitting in the stacked ensemble. This requires training k versions of each model type, where k is the number of jackknife splits.

Our approach is most similar to Martins et al. (2017), the major differences are: we do not use any internal features from the original MT system, and we do not need to "jackknife" in order to create a stacked ensemble. Using only NMT with attention, we are able to surpass the state-of-the-art in APE and match it in QE.

4.3.1 Factored Inputs for seq2seq

Alexandrescu and Kirchhoff (2006) introduced linguistic factors for neural language models. The core idea is to learn embeddings for linguistic features such as part-of-speech

(POS) tags and dependency labels, augmenting the word embeddings of the input with additional features. In fact, we have already used this idea in the first part of this chapter, when embeddings for source and target words were concatenated together to create the input for the BBLM. Recent work has shown that NMT performance can also be improved by concatenating embeddings for additional word-level factors to source-word input embeddings (Sennrich and Haddow, 2016). The input representation e_j for each source input x_j with factors F thus becomes Eq. 4.2:

$$e_j = \Big\|_{k=1}^{|F|} \mathbf{E}_k x_{jk} \tag{4.2}$$

where $||$ indicates vector concatenation, $\mathbf{E}_k$ is the embedding matrix of factor k, and x_{jk} is a one hot vector for the k-th input factor. This approach to adding additional feature representations into seq2seq models is very flexible, and enables us to augment the token-level input representations used for NMT with features such as POS tags or dependency parse information, which are known to improve the performance of QE models.

4.3.2 Neural Models for APE and word-level QE

In this section we describe five model types used for APE-QE, each based upon a different input representation, as well as ensembles of these models which turn out to be the best-performing overall. We design several features to be included as inputs to APE. The operating hypothesis is that that features which have proven useful for QE should also have a positive impact upon APE performance.

Our baseline models are the same models used in Junczys-Dowmunt and Grundkiewicz (2016) for the English–German APE task at WMT 2016.[10] The authors provide trained $SRC \rightarrow PE$ and $MT \rightarrow PE$ models, which correspond to the last four checkpoints from fine-tuning the models on the 500,000 segments of synthetic training data concatenated with the task-internal APE data up-sampled 20 times. These models are referred to as *SRC* and *MT*.

[10]These models have been made available by the authors at `https://amunmt.github.io/examples/postedit/`

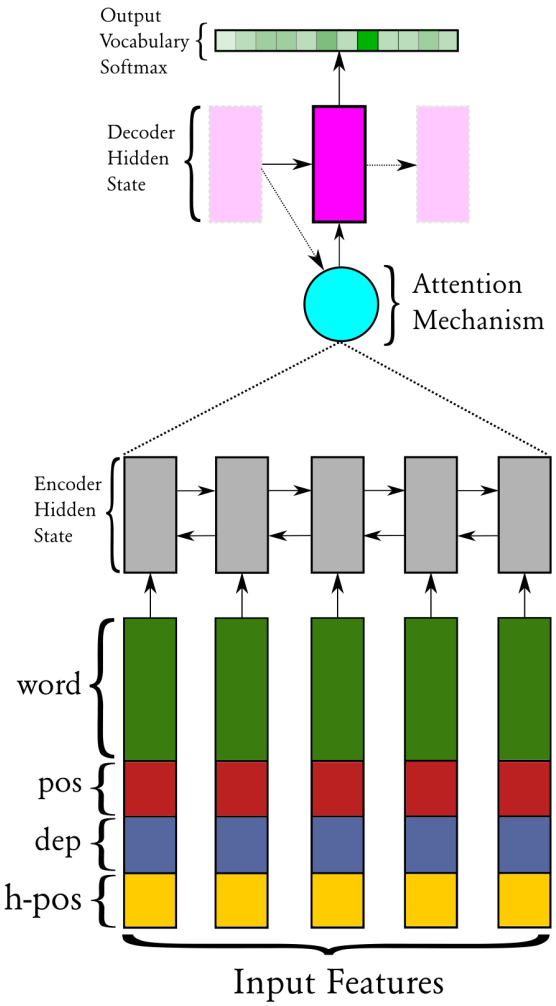


Figure 4.8: Schematic of the architecture of our factored NMT systems

Adding Word Alignments

Previous work has shown that alignment information between source and target is a critical component of current state-of-the-art word level QE systems (Kreutzer et al., 2015a; Martins et al., 2016). The sequential inputs for structured prediction, as well as the feed-forward and recurrent models in existing work obtain the source-side features for each target word using the word-alignments provided by the WMT task organizers. We have already pointed out this information is not likely to be available in many real-world use cases, and that the use of this information also means that the MT system used to produce the hypotheses is not actually a "black box", which is part of our definition of the QE task.

Because our models rely upon synthetic training data, and because we wish to view the MT system as a true black-box, we instead use the SRC NMT system to obtain the alignment features. The attention model for NMT produces a normalized vector of weights at

each timestep, where the weights can be viewed as the "alignment probabilities" for each source word (Bahdanau et al., 2014). In order to obtain the input representation shown in table 4.10, we use the source word with the highest weight from the attention model as an additional factor in the input to another MT-aligned → PE system. This *MT-aligned* → PE system thus depends upon the SRC → PE system to produce the additional alignment factor.

Inputting Both Source and Target

Following Crego et al. (2016a), we train another model which takes the concatenated source and MT as input. The two sequences are separated by a special *BREAK* token. We refer to this system as *SRC+MT*.

Adding Part-of-Speech and Dependency Features

Sennrich and Haddow (2016) showed that information such as POS tags, entity labels from named entity recognition (NER) systems, and syntactic roles can be included in the input to NMT models, generally improving performance. Inspired by this idea, and noting the importance of these features for word-level QE models, we select some of the top performing features from Martins et al. (Martins et al., 2016), and include them as input factors to create the *SRC+MT-factor* model. The base representation is the concatenated SRC+MT (again with a special *BREAK* token). For each word in the English source and the German hypothesis, we obtain the part-of-speech tag, the dependency relation, and the part-of-speech of the head word, and include these as input factors. For both English and German, we use spaCy[11] to extract these features for all training, development, and test data. The resulting model is illustrated in figure 4.8.

Extending Factors to Sub-word Encoding

Our NMT models use subword encoding (Sennrich et al., 2016b), but the additional factors are computed at the word level. Therefore, the factors must also be segmented to match the BPE segmentation. We use the {BILOU}-[12] prefixes common in sequence-labeling tasks such as NER to extend factor vocabularies and map each word-level factor to the sub-word segmentation of the source or target text.

[11] https://spacy.io/

[12] **B**egin, **I**nside, **L**ast, **O**utside, **U**nit

Table 4.10 shows the input representations for each of the model types using an example from the WMT 2016 test data.

Ensembling NMT Models

We average the parameters of the four best checkpoints of each model type, and create an ensemble of the resulting five models, called *Avg-All Baseline*. We then tune this ensemble for TER (APE) and $\mathrm{F1}_{\mathrm{Mult}}$ (QE), using MERT (Och, 2003). The tuned models are called *Avg-All APE-Tuned* and *Avg-All QE-Tuned*, respectively. After observing that source-only models have the best single-model QE performance (see section 4.3.4), we created a final $\mathrm{F1}_{\mathrm{Mult}}$ tuned ensemble, consisting of the four individual SRC models, and the averaged models from each other type (an ensemble of eight models total), called *4-SRC+Avg-All QE-Tune*.

Tuning

Table 4.9 shows the final weights for each ensemble type after tuning. In line with the two-model ensemble presented in Martins et al. (2017), tuning models for F1-Mult results in much more weight being allocated to the SRC model, while TER tuning favors models with access to the MT hypothesis.

	APE (TER)	QE (F1-Mult)
SRC	.162	.228
MT	.003	-.183
MT-aligned	.203	.229
SRC+MT	.222	.231
SRC+MT-factor	.410	.129

Table 4.9: Final weights for each model type after 10 iterations of MERT for tuning objectives TER and F1-Mult.

4.3.3 APE-QE Experiments

All of our models are trained using Nematus (Sennrich et al., 2017a). At inference time we use our own decoder, which supports weighted log-linear ensembles of Nematus models[13]. Following Junczys-Dowmunt and Grundkiewicz (2016), we first train each model type on the large (4 million instances) synthetic training data, then fine tune using the

[13] https://github.com/chrishokamp/constrained_decoding

SRC	auto vector masks apply predefined patterns as vector masks to bitmap and vector objects .
MT	automatische Vektor- masken vordefinierten Mustern wie Vektor- masken , Bitmaps und Vektor-objekte anwenden .
MT-aligned	automatische|auto Vektor-|vector masken|masks vordefinierten|apply Mustern|patterns wie|as Vektor-|vector masken|masks ,|to Bitmaps|to und|and Vektor-|vector objekte|objects anwenden|apply .|.
SRC+MT	auto vector masks apply predefined patterns as vector masks to bitmap and vector objects . BREAK automatische Vektor- masken vordefinierten Mustern wie Vektor- masken , Bitmaps und Vektor- objekte anwenden .
SRC+MT Factored	Auto|JJ|amod|NNS vector|NN|compound|NNS masks|NNS|nsubj|VBP apply|VBP|ROOT|VBP predefined|VBN|amod|NNS patterns|NNS|dobj|VBP as|IN|prep|NNS vector|NN|compound|NNS masks|NNS|pobj|IN to|TO|aux|VB bitmap|VB|relcl|NNS and|CC|cc|VB vector|NN|compound|NNS objects|NNS|conj|VB .|.|punct|VBP BREAK|BREAK|BREAK|BREAK Automatische|ADJA|nk|NN Vektor-|B-NN|B-sb|B-VVINF masken|I-NN|I-sb|I-VVINF vordefinierten|ADJA|nk|NN Mustern|NN|pd|NN wie|KOKOM|cd|NN Vektor-|B-NN|B-cj|B-KOKOM masken|I-NN|I-cj|I-KOKOM ,|$,\|punct\|NN Bitmaps\|NN\|cj\|NN und\|KON\|cd\|NN Vektor-\|B-NN\|B-cj\|B-KON objekte\|I-NN\|I-cj\|I-KON anwenden\|VVINF\|ROOT\|VVINF .\|$.|punct|VVINF
PE (Reference)	Automatische Vektormasken wenden vordefinierte Mustern als Vektormasken auf Bitmap- und Vektorobjekte an .
Gold Tags	OK OK BAD OK BAD OK BAD BAD OK OK BAD OK

Table 4.10: Examples of the input for the five model types used in the APE and QE ensembles. The pipe symbol '|' separates each factor. '-' followed by whitespace indicates segmentation according to the subword encoding.

500,000 instance synthetic dataset, concatenated with the task-internal training data up-sampled 20 times. Finally, for *SRC+MT* and *SRC+MT-factor* we continued fine-tuning each model for a small number of iterations using the min-risk training implementation available in Nematus, with BLEU score as the objective (Shen et al., 2016). Table 4.11 shows the best dev result after each stage of training.

Model	General	Fine-tune	Min-Risk
MT-aligned	60.31	67.54	–
SRC+MT	59.52	68.68	69.44
SRC+MT-factor	57.59	68.26	69.76

Table 4.11: Best BLEU score on dev set after each of the training stages. *General* is training with 4M instances, *Fine-tune* is training with 500K + upsampled in-domain data, *Min-Risk* uses the same dataset as *Fine-tune*, but uses a minimum-risk loss with BLEU score as the target metric.

For both APE and QE, we use only the task-specific training data provided for the WMT 2017 APE task, including the extra synthetic training data[14]. However, note that the SpaCy models used to extract features for the factored models are trained with external data – we only use the off-the-shelf models provided by the SpaCy developers.

To convert the output sequence from an APE system into $\mathrm{OK}, \mathrm{BAD}$ labels for QE, we

[14] http://www.statmt.org/wmt17/ape-task.html

use the APE hypothesis as a pseudo-reference, which is then aligned with the original MT hypothesis using TER (Snover et al., 2006a), as explained at the beginning of this section.

4.3.4 APE-QE Results

WMT 2016 Dev

Model Input	BLEU	TER ↓	F1-Mult	Accuracy
WMT 16 Best	68.94	.215	.493	–
Martins et al (2017)	–	–	**.568**	–
SRC	55.47	.315	.506	.803
MT	66.66	.232	.328	.834
MT-aligned	68.32	.215	.437	.852
SRC+MT	69.17	.211	.477	.857
SRC+MT-factor	69.75	.209	.484	.859
Avg-All Baseline	71.02	.199	.476	.862
Avg-All APE-Tune	**71.22**	**.197**	.510	**.866**
Avg-All QE-Tune	66.92	.228	.554	.857
4-SRC+Avg-All QE-Tune	67.16	.225	.567	.860

WMT 2016 Test

Model Input	BLEU	TER ↓	F1-Mult	Accuracy
WMT Baseline	62.11	.248	.324	–
WMT 16 Best	67.65	.215	.493	–
Martins et al (2017)	67.62	.211	**.575**	–
SRC	55.58	.304	.519	.809
MT	65.85	.234	.347	.837
MT-aligned	67.69	.216	.447	.854
SRC+MT	68.03	.212	.477	.857
SRC+MT-factor	68.28	.211	.473	.857
Avg-All Baseline	**70.05**	.198	.492	.865
Avg-All APE-Tuned	70.04	**.196**	.516	**.868**
Avg-All QE-Tuned	66.93	.219	.573	.864
4-SRC+Avg-All QE-Tune	66.94	.219	**.575**	.865

Table 4.12: Results for all models and ensembles on WMT 16 development and test datasets

Table 4.12 shows the results of our experiments using the WMT 16 development and test sets. For each system, we measure performance on BLEU and TER, which are the metrics used in APE task, and also on $\mathrm{F1}_{\mathrm{Mult}}$, which is the primary metric used for the word level QE task. Overall tagging accuracy is included as a secondary metric for QE.

Tables 4.13 and 4.14 show the official results of the 2017 word level QE and APE shared

Model	F1-Mult	F1-BAD	F1-OK
• POSTECH/Combined-Multilevel-Ensemble	0.581	0.637	0.913
• DCU/SRC-APE-QE-TUNED	0.575	0.627	0.917
• DCU/AVG-ALL	0.573	0.625	0.917
POSTECH/SingleLevel-Ensemble	0.561	0.619	0.906
• Unbabel/ensemble	0.495	0.560	0.885
Unbabel/linear	0.463	0.529	0.875
UGENT-LT3/SCATE-RF	0.411	0.492	0.836
CDACM/RNN	0.391	0.49	0.833
UGENT-LT3/SCATE-ENS	0.381	0.464	0.821
POSTECH/WORD-RNN-QV3	0.380	0.447	0.850
POSTECH/WORD-RNN-QV2	0.376	0.454	0.828
UAlacant/SBI-Online-baseline	0.367	0.456	0.805
Baseline 2017	0.360	0.404	0.892

Table 4.13: Official WMT 17 EN-DE Word Level QE task results. Our submissions are labeled "DCU/SRC-APE-QE-TUNED" and "DCU/AVG-ALL". Winning entries are marked with a • and are statistically significantly different from all others. WMT 16 submissions are highlighted with cyan, and we list only the systems which outperformed the WMT 2017 baseline.

tasks, as published in Bojar et al. (2017). In the QE task, both of our submissions were ranked first (note bullet points indicating statistical significance). In the APE task, our submissions placed fourth and fifth, but it is worth noting that the higher-ranked systems both were purpose-built for APE, and did not also submit to the QE task.

All systems with input factors significantly improve APE performance over the baselines. For QE, the trends are less clear, but point to a key difference between optimizing for TER vs. $\mathrm{F1}_{\mathrm{Mult}}$: optimization for $\mathrm{F1}_{\mathrm{Mult}}$ probably lowers the threshold for changing a target word, as opposed to copying it from the MT hypothesis. This hypothesis is supported by the observation that the source-only APE system outperforms all other single models on the QE metrics. Because the source-only systems cannot resort to copying words from the input, they are forced to make the best guess about the final output, and words which are more likely to be wrong are less likely to be present in the output. If input factors were used with a source-only APE system, the performance on word-level QE could likely be further improved. However, this hypothesis needs more analysis and experimentation to confirm.

ID	Avg. TER	BLEU
FBK Primary	19.6	70.07
AMU Primary	19.77	69.5
AMU Contrastive	19.83	69.38
DCU Primary	20.11	69.19
DCU Contrastive	20.25	69.33
FBK Constrastive	20.3	69.11
FBK USAAR Contr.	21.55	67.28
USAAR Primary	23.05	65.01
LIG Primary	23.22	65.12
JXNU Primary	23.31	65.66
LIG Contrastive-Forced	23.51	64.52
LIG Contrastive-Chained	23.66	64.46
CUNI Primary	24.03	64.28
USAAR Contrastive	24.17	63.55
Baseline	24.48	62.49
(Simard et al., 2007)	24.69	62.97
CUNI Contrastive	25.94	61.65

Table 4.14: Official WMT 17 EN-DE Word Level APE task results. Our submissions are labeled "DCU Primary" and "DCU Contrastive"

4.4 Discussion

This chapter has given an overview of our main contributions to word level QE, which include Marmot, a library which provides a widely-used baseline for training and evaluating QE models, investigations into novel deep recurrent architectures for word level QE, and a new formulation of neural automatic post-editing, targeted at solving both APE and word level QE with the same suite of models.

In particular, the APE-QE approach unifies models for APE and word-level QE by leveraging the flexibility of NMT, and achieved results which are on-par with more complex, purpose-built systems, as well as achieving the objective of simplifying the complex feature extraction, training, and inference pipelines discussed throughout the chapter. We extended the ideas of Junczys-Dowmunt et al. (2016) and Martins et al. (2017) by focusing upon including QE-specific features as factors to NMT models for APE, and by ensembling NMT models with different factored input representations, and tuning for either APE or QE. We were thus able to achieve competitive results in both tasks. The complementary nature of these tasks points to future avenues of exploration, such as joint

training using both QE labels and reference translations, as well as the incorporation of even more QE features as input factors.

Our best APE-QE system was successfully integrated into a real-time QE server, which was a key component of an interactive post-editing CAT interface prototype. The design and details of this system are presented in chapter 7.

Chapter 5

Modeling Interactive Neural Machine Translation

5.1 Introduction

In the interactive scenarios discussed in chapter 2, we envisioned a translation framework where a human translator cooperates with an MT system while translating, taking advantage both of the human's ability to precisely judge translation quality, and of the machine's ability to quickly find candidate translations for any source-language input. However, prefix-based IMT systems to date have not fully achieved this goal, because the output of the system does not include information about the model's confidence that its prediction is good.

Therefore, in current IMT interfaces, the translator is effectively wearing two hats: on one hand they are reading MT output, and searching for errors, and on the other hand they are composing parts of translations from left-to-right. This setting is basically the same as the text prediction scenario discussed in chapter 2, and was investigated in the first user study presented in chapter 3.

Unlike the auto-complete models discussed in chapter 3, an IMT system can predict translations of any length, from a complete target output when the user-supplied prefix is empty, to a single end-of-sentence (EOS) token in cases where the user-provided prefix is a complete translation. Intuitively, an IMT model should be most confident when it is predicting the word directly after the end of the input prefix, and uncertainty about the

continuation of the translation should increase as the predicted suffix becomes longer[1]. For certain inputs, the model may be able to predict a few words with high certainty, but may then reach a point where its confidence about the continuation becomes low. Low confidence about the continuation of a translation should correspond to a high probability that the translator will intervene at this point, in order to fix an error. Therefore, we would like to do better than simply predicting the full suffix for every given prefix, and allow the MT model to truncate its output at the point where it is not confident, delegating control back to the translator.

This chapter presents a series of experiments evaluating the use of predicted model confidence to optimize the utility of IMT model outputs. We focus upon *RQ3*, but leave our proposal for a generalized framework beyond prefix-based IMT for chapter 6. The ultimate goal of the confidence modeling techniques presented in this chapter is to reduce the cognitive load on the translator by removing suffixes that are likely to be wrong, and would need to be deleted or revised. We can assess the hypothetical ability of a hypothesis truncation method to save translator effort using the metrics introduced in section 2.4.5 or by simply counting the input operations that would be saved if a translator were using the IMT system instead of translating from scratch.

We also implement a simple means of converting a NMT system into an interactive NMT (I-NMT) system, in order to explore the training of integrated models of confidence for IMT. Taking advantage of the structural characteristics of standard encoder-decoder models, we can easily convert an NMT model into a model supporting prefix decoding, an idea that has been explored by several recent works (Wuebker et al., 2016; Knowles and Koehn, 2016; Peris et al., 2017). We also implement this simple conversion of NMT to I-NMT, and further extend this architecture to support integrated models of translation confidence.

Most of the empirical results in this chapter are negative in the sense that they do not improve performance over simpler baselines. However, the experiments and model architectures designed during this portion of the thesis provided direct inspiration for the ideas in chapter 6, which is a powerful framework for interactive translation beyond prefix-decoding.

[1] some of the results in this chapter call this intuition into question

The chapter is organized as follows: section 5.2 explains how encoder-decoder based NMT systems can be easily re-purposed for interactive scenarios, section 5.3 proposes the use of internal features from NMT systems as indicators of model confidence, applying this information to truncate hypotheses when an IMT model is not confident that its prediction will be correct. Section 5.4 investigates explicitly *learning* auxiliary models of confidence for interactive translation. Finally, section 5.5 discusses the main conclusions learned from our work on IMT confidence models and prefix attention models for IMT.

5.2 Interactive Neural Machine Translation

As discussed in section 2.4.1, NMT is well-suited to interactive scenarios, because the structure of the NMT decoder already factorizes the probability of a given target output into an auto-regressive computation from beginning to end. Thus, in order to use an encoder-decoder model of any sort to enable one of the interactive settings described in section 2.4, we simply need to modify the computational pipeline of the decoder to compute the representation of the target prefix before computing the most probable output suffix. Note that this basic re-purposing of the model for interactivity does not require any changes or updates to the trained NMT model's parameters – a model trained in the standard manner using teacher-forcing with parallel source and target-language segments may be reused directly fpr I-NMT.

In order to convert an NMT model into an I-NMT model, we begin by modifying the architecture to take two inputs: a complete source sequence, and an empty, partially complete, or completed target sequence. The system then generates the target language sequence which, when concatenated onto the provided prefix, forms a complete translation of the source. In case the prefix is empty, the task is equivalent to the standard machine translation problem. If the prefix is a completed translation, the system should only predict the EOS token, i.e. the model should be able to detect when a translation is already complete.

The decoder component of the model is driven by a recurrent transition which begins with an initial state $\mathbf{h}_0$, which our model computes by performing a parameterized linear transformation of the final source state, followed by a non-linear function, i.e. $\mathbf{s}_0 =$

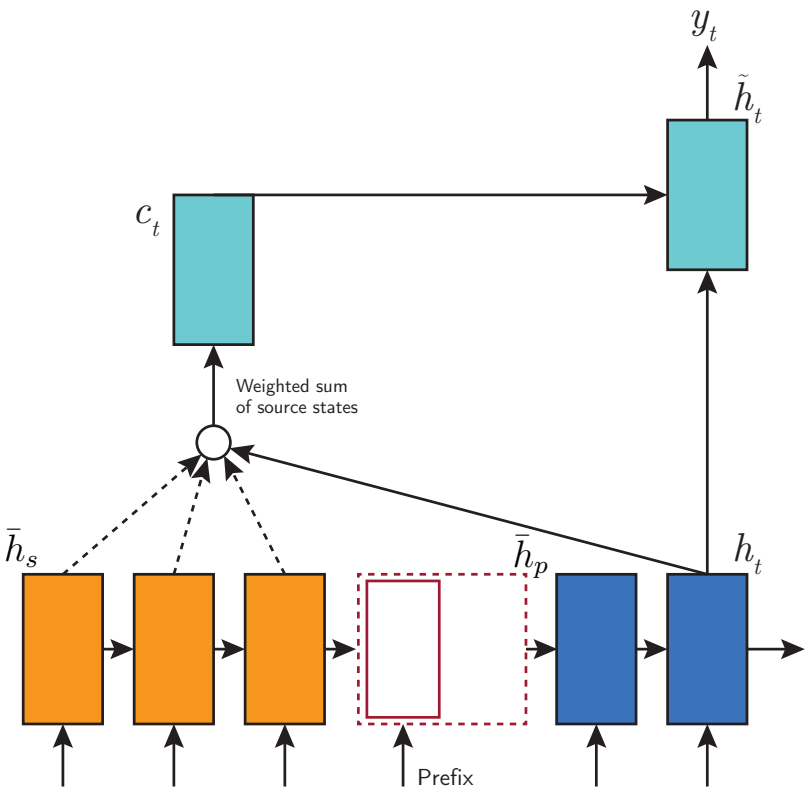


Figure 5.1: Attention-based NMT with prefix decoding. $\bar{h}_s$ is the sequence of source states, $\bar{h}_p$ is the sequence of prefix states, which are initialized with the final states of the source representation. The prefix h_p is first encoded by the target model, computing attention at each time step. The suffix representation is then initialized with the final state of the prefix sequence, and the model is asked to generate a suffix.

$\tanh(\mathbf{W}_s\mathbf{h}_S)$. The key modification that needs to be done to convert an NMT system into an IMT system is that the states for each item in the prefix must first be encoded by the decoder beginning with the initial state $\mathbf{s}_0$, then the final state of the prefix $\mathbf{h}_P$ must be used as the initial state for the suffix hypothesis. Initializing the decoder transition with the final state of the prefix corresponds to the prefix decoding discussed in section 2.4.1. Figure 5.1 visualizes an encoder-decoder I-NMT model with attention.

5.2.1 Accounting for Unknown Words

As discussed in section 2.4.1, an IMT system must first determine which portion of the source has already been translated by the prefix in order to determine how translation should continue. In SMT scenarios, the prefix must be "force-decoded" (Yu et al., 2013; Huang et al., 2012), meaning that the derivations from the search algorithm are constrained to those which match the target prefix.

Ideally, IMT systems must be able to account for the case where a user-provided token or phrase in the prefix is not found in the phrase table. There are two main approaches to this issue which have been explored: (1) perform fuzzy matching by searching for the node in the output lattice which has lowest edit-distance to the prefix (Ueffing and Ney, 2005; Casacuberta et al., 2014), or (2) dynamically add synthetic phrase pairs to the phrase table, which contain all possible alignments between unknown target words and source words (Green, 2014).

Sub-word MT The first NMT systems dealt with unknown symbols in source and target sequences by mapping them to a special UNK token. Sennrich et al. (2015) introduced a pre-processing methodology which produces a sub-word vocabulary which can cover all source and target strings without the need for a special token. This approach, and variants such as *wordpiece* encoding (Wu et al., 2016), are now state of the art for NMT for most language pairs. For IMT in particular, sub-word encoding provides an elegant means to avoid any heuristically motivated solution to unknown words in the user-provided prefix. All of the models presented in the following sections use sub-word encoding, so the models can predict a suffix for any input, as long as the individual *characters* used in the input were observed in the training data.

5.3 Built-in Confidence Measures in NMT Systems

The NMT model with attention architecture proposed by Bahdanau et al. (2014) contains two softmax computations: the *output softmax*, and the *attention softmax*. The softmax function can be seen as converting a vector of unnormalized scalars into a multinomial probability distribution. In this view, the vector component with the highest weight is the most probable, and the distribution of probability mass across the components is an indication of a model's confidence. Indeed, this interpretation underlies all likelihood-based optimization, and has also been used to decipher the impact of different inputs on the predictions made by deep learning models (Ribeiro et al., 2016). We would like to measure the extent to which we can use these built-in confidence measures to optimize IMT prediction length. In particular, we investigate how to leverage the distribution of weights from the NMT attention mechanism to make decisions about the optimal length

of IMT hypotheses.

At inference time, an NMT system chooses a token y_i to output at each timestep t. This is done by choosing the index of $\mathbf{y} \in \mathbb{R}^{|V|}$ with the maximum value, where $|V|$ is the target vocabulary size. The probability of predicting each y_i at timestep t depends on the previous output, y_{t-i}, the current state s_t, and a representation of context c_t, which are combined together by a function $g()$, as shown in Eqs. 5.1 and 5.2[2]:

$$p(y_t|y_1, \ldots, y_{t-1}, \mathbf{x}) = g(y_{t-1}, s_t, c_t) \tag{5.1}$$

$$s_t = f(s_{t-1}, y_{t-i}, c_t) \tag{5.2}$$

where $f()$ is a sub-network which takes the previous state s_{t-1}, the embedding of the previously predicted token y_{t-1}, and a vector representing the context c_i, and outputs the current state of the model s_t. In practice, $g()$ is a sub-network consisting of one or more fully-connected layers, with a final linear transformation that maps the vectors of the internal size of the model into the size of the target vocabulary, Eq. 5.3:

$$\mathbf{h}^{\text{out}} = \mathbf{W}_{\text{out}}\mathbf{h} \tag{5.3}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{M \times |V|}$, and M is the size of the model's hidden states. Note the lack of a non-linear transformation such as $\tanh()$ and the lack of a bias term $\mathbf{b}$ in the $\mathbf{W}_{\text{out}}$ transformation. The output transformation results in a vector of scalars which we refer to as $\mathbf{h}^{\text{out}}$. Finally the softmax function is used to convert the unnormalized $\mathbf{h}^{\text{out}}$ into a distribution over the target vocabulary – we thus refer to Eq. 5.4 as the *output* softmax:

$$p(y_t|y_1, \ldots, y_{t-1}, \mathbf{x}) = \frac{\exp(\mathbf{h}^{\text{out}}_i)}{\sum_{j=1}^{|V|} \exp(\mathbf{h}^{\text{out}}_j)} \tag{5.4}$$

The *attention* softmax, on the other hand, results in a probability distribution over the encoder's hidden states $\mathbf{h}$, which is used to compute a weighted sum over $\mathbf{h}_j$, at timestep t resulting in the c_t term in equations 5.1 and 5.2 (Eq. 5.5):

[2]Notation modeled after Bahdanau et al. (2014)

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \tag{5.5}$$

The attention weights α of the model at each time step are computed as:

$$\alpha_i = \frac{\exp(\mathbf{e}_i)}{\sum_{k=1}^{T_s} \exp(e_{ik})} \tag{5.6}$$

$$\mathbf{e}_i = \text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) \tag{5.7}$$

Where score is the alignment function from Bahdanau et al. (2014), which computes each element e_{ij} as in equation 5.8:

$$e_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{h}_t + \mathbf{U}_a \mathbf{h}_s) \tag{5.8}$$

See Bahdanau et al. (2014) and Luong et al. (2015) for more details on possible variants of the attention model. In the following sections we show how the alignment weights α_i can be used at each timestep to decide whether to truncate an IMT hypothesis at the point where alignment confidence becomes low. Our operating hypothesis is that the model is less confident about its prediction when the probability mass of the attention softmax is spread over many source tokens, instead of focused upon a single token.

5.3.1 Using Attention Weights to Determine Model Confidence

As discussed in section 5.3, at each timestep, the decoder computes a vector of alignment weights α_i where each α_{ik} can be interpreted as the probability that target word i is aligned with source word k. Inspired by Ueffing and Ney (2005), who used IBM Model 1 weights as a confidence metric to prune a lattice output by an SMT decoder, we propose searching for the best ad-hoc threshold γ of the minimum alignment weight, and truncate each hypothesis at the first location where a target word is not aligned to a source word with weight $\geq \gamma$. Intuitively, we are assuming that when the attention model cannot concentrate enough probability mass onto a single token, its confidence about the output prediction should also be low.

Dataset	BLEU	METEOR
newstest 2014	18.11	.396
newstest 2015	20.00	.401

Table 5.1: NMT system baseline performance on WMT newstest datasets

Dataset	IMT F1	Suffix BLEU	Suffix METEOR	Total Instances
newstest2014-500	.163	19.33	.477	12811
newstest2015-500	.169	20.44	.472	12943

Table 5.2: Neural IMT system prediction performance evaluated against reference suffixes created from newstest*-500 datasets. The **IMT F1** metric was defined in section 2.4.5. Suffix BLEU and suffix METEOR are computed by comparing the IMT generated suffixes to the reference suffix for every possibile (prefix, suffix) pair in each of the 500 instances.

5.3.2 Experiments

We used a modified version of attention-based NMT with global attention (Bahdanau et al., 2014; Luong et al., 2015) to build a neural IMT system in the manner described in section 5.2, by modifying a NMT model created with Blocks (van Merriënboer et al., 2015). The suffix truncation technique proposed in the previous section 5.3.1 was tested with an English–German attention-based model where the implementation exactly follows the equations in section 5.3.

Configuration details The NMT model was trained with the 2015 WMT English-German parallel training data (Bojar et al., 2015) concatenated with the WIT corpus (Cettolo et al., 2012), and mapped into a combined En–De sub-word encoding with 90000 symbols (Sennrich et al., 2015). The system is trained for up to 5 epochs — the best model was captured by performing BLEU score validation on the newstest 2013 dataset each time the model has ingested 50,000 training examples. Table 5.1 shows the performance of the system on two test datasets, and confirms that our system is close to the state-of-the-art for non-ensembled models. The trained weights from this system were then used to initialize the I-NMT model in order to obtain predictions. Table 5.2 shows the I-NMT system's performance on several metrics, confirming that its performance is reasonable as well.

System	IMT F1
Baseline	.172
Threshold	.199
Cutoff	.208
Cutoff+Threshold	**.209**
Best Possible	.312

Table 5.3: Baseline IMT system performance on newstest2015_500, compared against various pruning heuristics. *Cutoff* indicates a hard global cutoff length of 3, which was chosen by tuning on held-out data. *Threshold* indicates a minimum alignment probability threshold for the best aligned token (.388), *Cutoff+Threshold* indicates first extending the suffix until the optimal cutoff, then continuing extension of the suffix until a predicted word falls below the alignment threshold. *Best Possible* is the maximum achievable score that an oracle method could achieve under the current model.

Test Dataset Preparation

To create the test datasets, we take each $(\text{source}, \text{target})$ pair in the original parallel development sets from WMT, and map each target sentence into every possible $(\text{prefix}, \text{suffix})$ pair. This results in $|\text{target}| + 1$ training instances for each parallel sentence pair, because we also consider the case where the prefix is empty (system should predict a complete hypothesis), and the case where the prefix is the complete target sequence (system should predict nothing, because the prefix is already complete).

Note that our method of expanding the each instance in test data into $(\text{source}, \text{prefix}, \text{suffix})$ triples increases the total number of test instances drastically, from $|\text{instances}|$ to $|\mathbf{words}| + |\mathbf{instances}|$. We thus use only the first 500 pairs from newstest 2014 and from newstest 2015 to create our test datasets, which already yields over 10,000 test instances. We refer to these datasets as newstest*-500. Table 5.2 shows results on these datasets using standard MT metrics to evaluate the quality of the predicted suffixes.

Tuning

The threshold γ was tuned on held-out data, over settings from $0.0, \ldots, 1.0$. Figure 5.2 plots the effect of varying this threshold on average IMT F1, precision, and recall. Around $\gamma = 0.39$ a small boost in IMT F1 to 0.21 is observed, otherwise the diverging precision and recall scores cause the IMT F1 score to hover around 0.19.

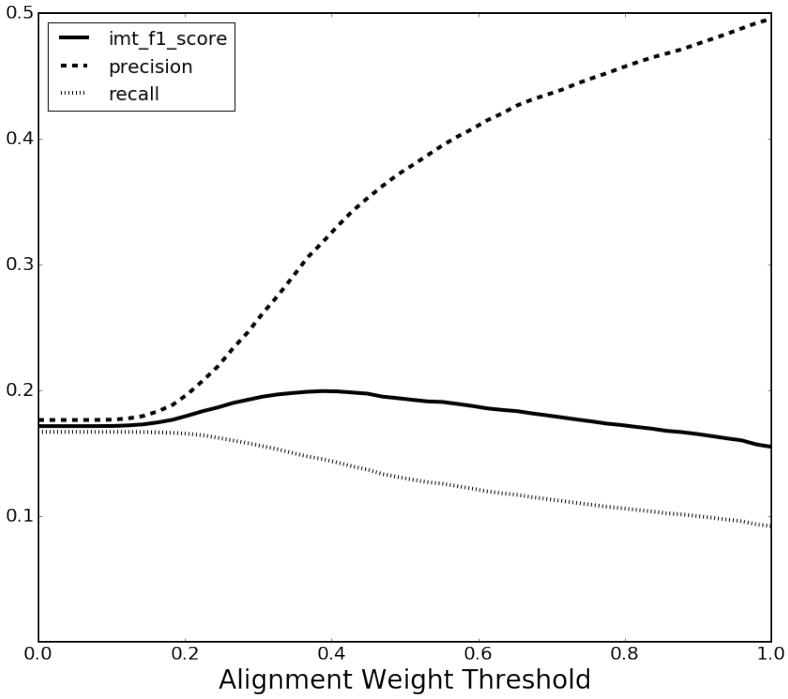


Figure 5.2: IMT F1, precision, and recall at different settings of alignment confidence threshold for newstest2015-500

5.3.3 Results

We consider two possible pruning heuristics according to IMT F1 (table 5.3); the final scores indicate the average performance over every test instance. The scores in table 5.3 were obtained by finding the optimal threshold and cutoff values using newstest2014_500, then computing the score at these thresholds for newstest2015500. We additionally report the maximum achievable IMT F1 using the reference suffix as an oracle. This oracle score is obtained by comparing the complete hypotheses from the IMT system with the references, and selecting the optimal cutoff for each hypothesis.

The results of these experiments indicate that simply cutting off hypotheses at a fixed point (optimal $\mathbf{cutoff} = 3$ in these experiments) is as effective as a more sophisticated use of attention weights as proxies to model confidence. The combination of cutoff heuristic and thresholding using attention weights obtains a very slight performance boost, but is not significant enough to warrant further investigation. However, two important conclusions can be drawn from these experiments: (1) truncating IMT hypotheses improves performance over simply predicting the complete suffix, and (2) an NMT model's internal confidence, in this case computed from the internal attention weights at each timestep, is

correlated with the probability that the predicted output word will be correct. Motivated by these insights, we designed a second set of experiments to test the use of a more sophisticated confidence model.

5.4 Learning Model Confidence

In this section, we propose a means of learning a model's confidence that its next prediction will be correct. This confidence score can be seen as an assessment degree of ambiguity at each decoding timestep. Our proposed model can be jointly trained together with a neural machine translation system, and then used to dynamically truncate the output of the I-NMT system's decoder at prediction time.

5.4.1 Auxilliary Confidence Models

Given an already trained model, we wish to leave the trained model parameters θ_{model} static, and train an auxiliary model to output the probability that the main model's output will be correct with respect to an unknown reference. In principle, this idea is very similar to confidence estimation and quality estimation; however, here we wish to use this information to truncate model predictions at the point where the auxiliary model predicts that the model has a low probability of being correct. In addition, unlike in the QE settings discussed in chapter 4, in this scenario we have access to the internal workings of the model, so we wish to build a model using "glass-box" features, instead of taking the "black box" approach discussed in chapter 4.

We propose using two sources of information to predict the model's confidence about its predictions: (1) the output softmax discussed in section 5.3, and (2) the internal state of the model computed by $g()$ (Eq. 5.2). In the following, we refer to the output of $g()$ at timestep t as $\mathbf{s}_t \in \mathbb{R}^{|hidden|}$, and to the output of Eq. 5.4 for every y_i at timestep t as $\mathbf{y}^t \in \mathbb{R}^{|V|}$.

Our confidence model is implemented as a three-layer fully-connected network, with a single sigmoid unit at the output; however, any model architecture is possible in principle. We refer to the parameters of the confidence model as $\theta_{\text{confidence}}$. Note that this set of parameters is disjoint from θ_{model}, so changing $\theta_{\text{confidence}}$ does not affect the predictions made by the I-NMT model. The value of the output is the probability that the predicted

token at the current timestep matches a the reference translation. This model can be trained with a binary cross-entropy loss: $\mathcal{L}_c$ (Eq. 5.9):

$$\mathcal{L}_c^{it} = -(c \log(\hat{c}) + (1 - c)(\log(1 - \hat{c})), \tag{5.9}$$

where we use $\mathcal{L}_c^{it}$ to denote the cross-entropy loss for instance i at timestep t. We use c to indicate the ground truth label (1 if the predicted word matches the reference word, 0 otherwise), and $\hat{c}$ to denote the confidence model's predicted label, to distinguish between the confidence model's predictions and the I-NMT model predictions $\hat{\mathbf{y}}$. Note that $\mathcal{L}_c$ is a function of the input $\mathbf{x}$, the user-prefix $\mathbf{p}$, the predicted suffix $\hat{\mathbf{y}}$ and the true suffix $\mathbf{y}$. The loss term $\mathcal{L}_c$ is the average cross-entropy over all training instances at all timesteps (Eq. 5.10):

$$\mathcal{L}_c = \frac{1}{N} \sum_{1}^{N} \frac{1}{T_i} \sum_{1}^{T_i} \mathcal{L}_c^{it}, \tag{5.10}$$

where T_i indicates the (suffix) length of training instance i, and N is the number of instances in the training dataset.

When training the auxiliary confidence model, only the parameters of the confidence model itself are updated; the I-NMT model's parameters are left unchanged — the gradient descent update at each training iteration t is computed as in Eq. 5.11:

$$\theta_{\text{confidence}}^{t+1} = \alpha \nabla_{\theta_{\text{confidence}}} \mathcal{L}_c + \theta_{\text{confidence}}^{t} \tag{5.11}$$

We have now described an auxiliary model that can be "attached" to an NMT or I-NMT system, and trained to predict the model's confidence about its outputs. The following section describes an experiment using this architecture.

Confidence Model Experiments

As mentioned above, we attempt to learn model confidence using a three-layer fully connected feed-forward neural network. We experiment with providing (1) the model hidden state $\mathbf{s}_t$ and (2) the softmax output as inputs to this model. The intermediate layers of the network are of size 300, and the output is a single sigmoid unit as described above.

Confidence Model Input	Input Operations Saved	Fraction of Total
random	5535	.031
$\mathbf{s}_t$	6216	.035
logits	6595	.037
$\mathbf{s}_t$ + **logits**	6596	.037
$\mathbf{s}_t$ + **logits** + dropout 0.2	**6615**	.037
oracle	11354	.063

Table 5.4: The raw number of input operations that would be saved by using each confidence model type to truncate hypotheses, along with the fraction of the total operations in the dataset that would account for. "random" indicates cutting off a hypothesis at a random point. Note that randomly cutting off hypotheses still would save some operations because many predictions do not match the reference, so cutting these off would keep the user from needing to delete these incorrect tokens.

The training and test datasets are created in the same manner as the test datasets described in section 5.3.2. In order to obtain the 1 or 0 label indicating whether each predicted token in a suffix is correct, we "force-decode" the reference suffix, and check whether the index with the maximum value in the model's softmax matches the reference token.

The value of the sigmoid output of the confidence model can be understood as the model's belief that the current predicted token will match the reference. We search for the optimal threshold of this value by tuning on held out data in the same manner as the experiment in section 5.3.2; however, instead of using IMT F1 as the tuning metric, we instead count the raw number of input operations that a hypothetical user would save when using this confidence threshold to truncate hypotheses. In order to compute the number of input operations, we compare the IMT generated suffix with the reference suffix, stopping at the point where the two suffixes do not match. If the IMT suffix continues beyond the point where it diverges from the reference, we assume the user would need to delete the part that doesn't match, so we decrease the operations saved by the number of tokens in the IMT suffix that do not match the reference[3]. For each word in the IMT suffix which matches the reference suffix, we increase the number of operations saved by 1.

Confidence Model Results

Table 5.4 presents the results of several configurations of the confidence model. By combining dropout with both the internal model state s_t, and the model softmax output, we obtain the best results. However, despite the fact that the confidence model is better than random, the total number of operations in the test set is 179008, much higher, meaning that our best model can only save a hypothetical user around 4% of the total operations. Therefore, we did not consider this a very promising result. The following section reviews the key takeaways from these experiments.

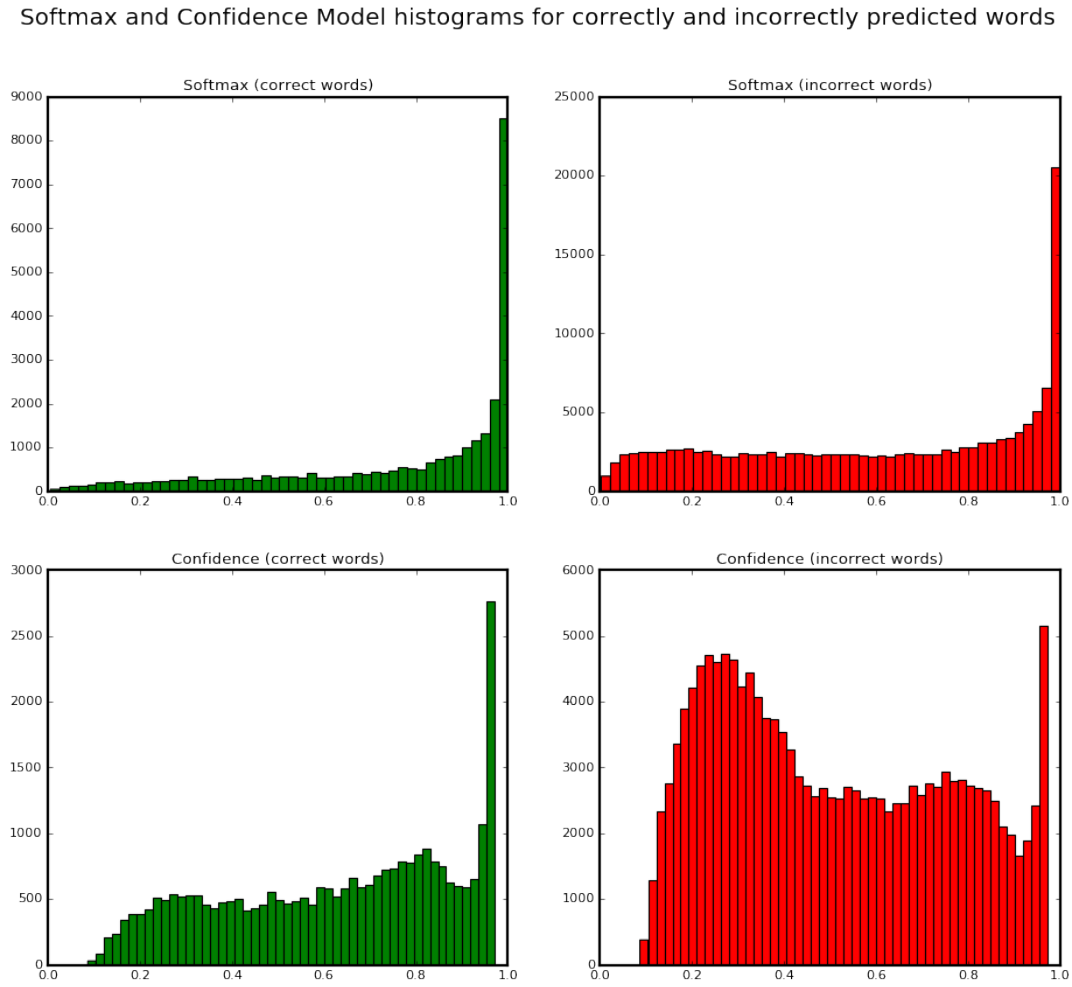


Figure 5.3: Histograms of values for correctly predicted tokens and incorrectly predicted tokens, for both the I-NMT softmax itself, and for the best confidence model.

Despite the somewhat discouraging empirical results of the experiments above, we conducted some additional analysis to try to better understand the confidence model's performance.

[3] Errors are counted from the first token which does not match, so a single incorrect word renders all following words incorrect

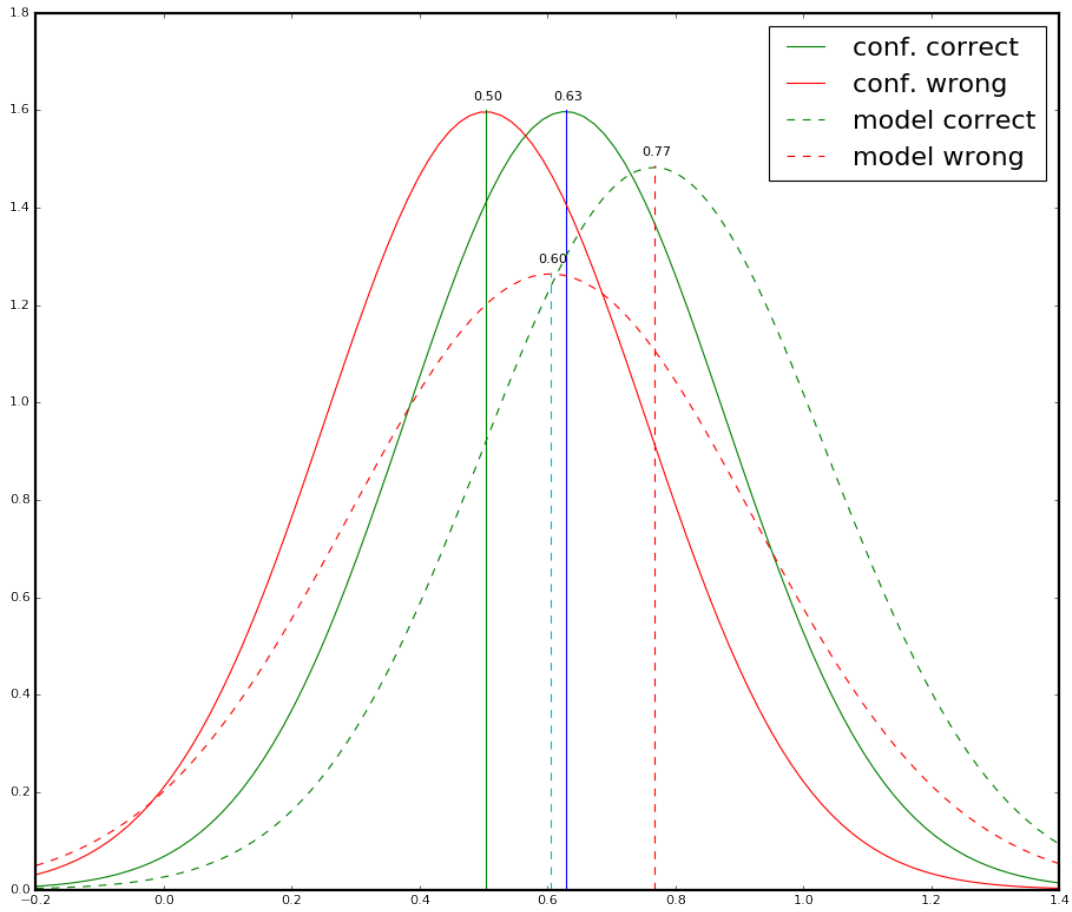


Figure 5.4: Gaussians of the values for correct and incorrect tokens, for the confidence model and for the N-IMT softmax.

Evaluating the Relationship between I-NMT Softmax and Confidence Model

As already mentioned, the values in the model softmax can be understood as a proxy to confidence, since we are mapping the output vocabulary of the model into a probability distribution, and choosing the maximum value at each timestep as our prediction. When our model puts most of the weight at a single index of the softmax, this can be interpreted as high confidence that the prediction is correct, whereas probability mass distributed over several indexes indicates more ambiguity about the correct token. We have already established that a confidence model with access only to the softmax outputs, and not to the internal state of the model does not perform as well as one with access to the internal state (Table 5.4). Figure 5.3 hints at why this may be the case. We see histograms of the softmax and confidence values for correctly and incorrectly predicted words. Note that the model softmax tends to be quite high, even for incorrectly predicted words, indicating

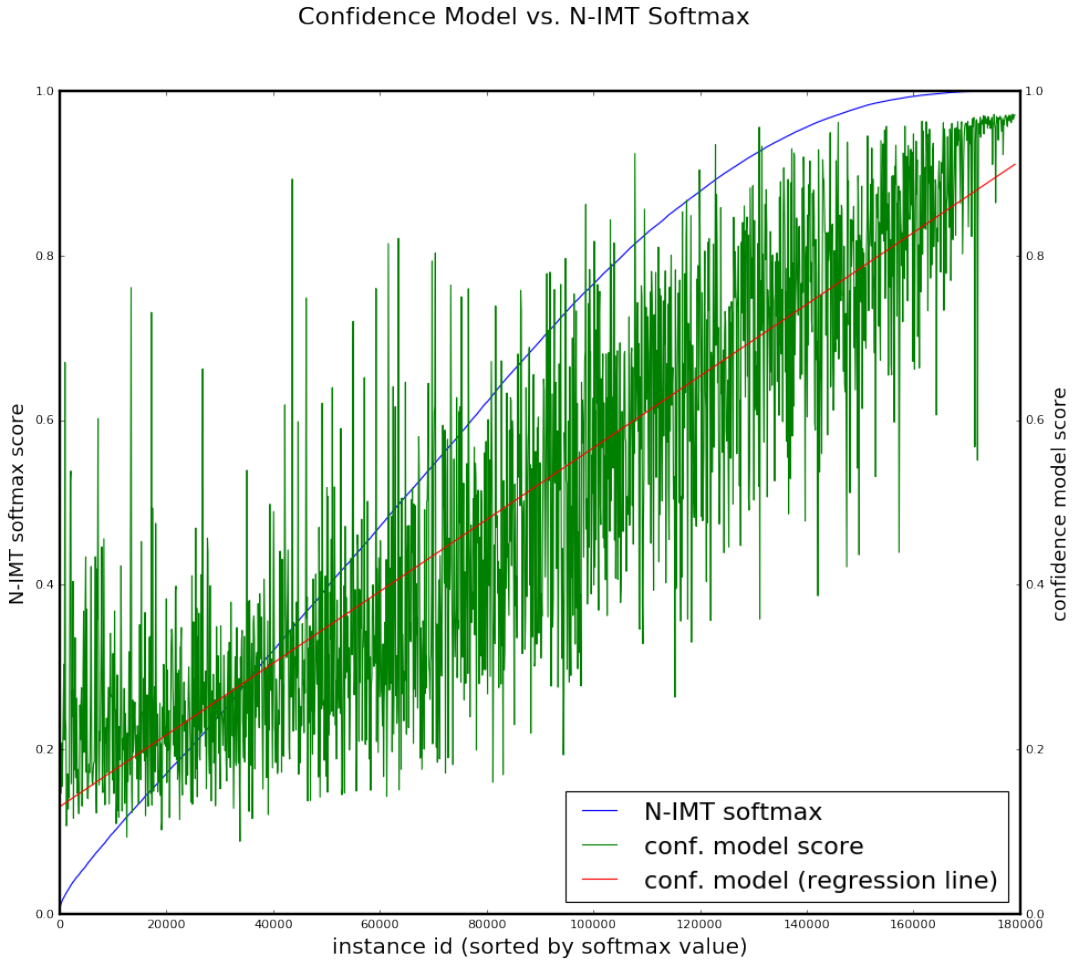


Figure 5.5: Visualizing the correlation between I-NMT softmax value and confidence model score.

that the I-NMT model itself is effectively "always confident" about its predictions. On the other hand, the confidence model's distribution is more in-line with our expectations – incorrectly predicted words tend to have lower confidence than correctly predicted ones.

Figure 5.4 plot the same data.[4] We do see that, for both the confidence model and the I-NMT softmax, there is a clear separation between correct and incorrect predictions, but still a very broad overlap between the two. Figure 5.5 plots the relationship between the softmax value and the confidence model score for the same data. We notice that the correlation between the softmax value and the confidence model prediction is quite high, but that the confidence model predictions are more evenly distributed across the range of possible values, whereas the softmax is skewed towards very confident predictions.

[4]Although this data is not Gaussian-distributed, plotting the distribution of values as Gaussian gives clear visual feed-back on the mean and variance of each grouping.

5.5 Discussion

This chapter has introduced a simple means of converting an NMT system into an I-NMT system, and several experiments with confidence models for IMT. After this work was completed, several researchers, including Wuebker et al. (2016), Knowles and Koehn (2016), and Peris et al. (2017) published similar methodologies for converting NMT systems into I-NMT systems, showing that this method is broadly applicable, and can be implemented in a straightforward way in a variety of frameworks. In this chapter, we have focused specifically upon confidence models for truncating I-NMT output — to the best of our knowledge, this idea has not been thoroughly explored by any researchers except Ueffing and Ney (2005), who introduced the idea. We believe that much more exploration on this topic could be done.

Despite the mostly negative results of the experiments presented in this chapter, we may still draw some useful conclusions from the work conducted here. The main takeaway from our experiments is that truncating hypotheses can help to protect translators from needing to read and then remove long, incorrect suffixes. However, how much impact this could have in practice would need to be measured with a carefully designed user study, such as the one presented in Green (2014). We also integrated our I-NMT models into HandyCAT, and showed the potential of I-NMT in demos such as Hokamp and Liu (2015), but we leave user studies of these models for future work.

Chapter 6

Lexically Constrained Decoding

In this chapter, we take a step back from the specialized domains of user interfaces for CAT, word-level QE, and IMT, that we have discussed in previous chapters, and consider a more general problem space: how can we condition a model which outputs sequences of text to include arbitrary lexical constraints in its output? In fact, we go a step further, and ask: how can we *force* a model to include such constraints? This is a deep question with implications far beyond MT alone. This chapter directly investigates *RQ3*, proposing an algorithm that enables interactive translation beyond prefix-based IMT.

The output of many natural language processing models is a sequence of text. Examples include automatic summarization (Rush et al., 2015), machine translation (Koehn, 2010a; Bahdanau et al., 2014), caption generation (Xu et al., 2015), and dialog generation (Serban et al., 2016), among others. In each of these settings, which are often instances of the seq2seq tasks discussed in section 2.3, our model receives some input $\mathbf{x}$, and is asked to output a sequence $\hat{\mathbf{y}} = \{y_0 \dots y_T\}$. A general way to chose the best $\hat{\mathbf{y}}$ from the exponentially many choices available is to search for the $\hat{\mathbf{y}}$ which maximizes $p_\theta(\mathbf{y}|\mathbf{x}) = \prod_t p(y_t|\mathbf{x}; \{y_0 \dots y_{t-1}\})$, where we use p_θ to indicate the model's distribution with parameters θ. The techniques discussed in this chapter apply to any model which conducts such a search.

In many real-world scenarios, we have access to additional information that could inform the search for the optimal output sequence at inference time. Humans can provide corrections after viewing a system's initial output, or separate classification models may be able to predict parts of the output with high confidence. When the domain of the in-

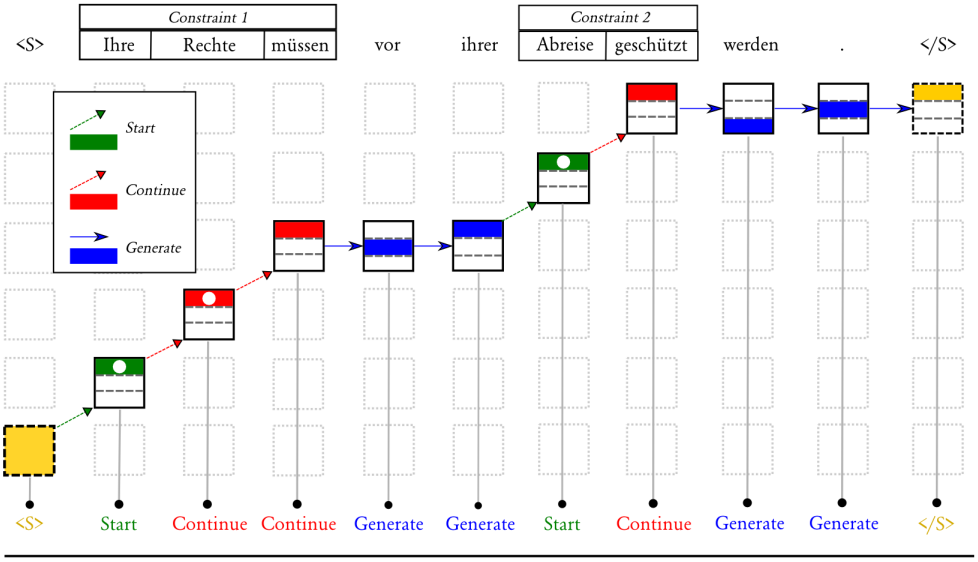


Figure 6.1: A visualization of the decoding process for an actual example from our English-German MT experiments. The output token at each timestep appears at the top of the figure, with lexical constraints enclosed in boxes. *Generation* is shown in blue, *Starting* new constraints in green, and *Continuing* constraints in red. The function used to create the hypothesis at each timestep is written at the bottom. Each box in the grid represents a beam; a colored strip inside a beam represents an individual hypothesis in the beam's k-best stack. Hypotheses with circles inside them are *closed*, all other hypotheses are *open*. (Best viewed in colour).

put is known at prediction time, a domain terminology or named-entity gazetteer may be employed to ensure specific phrases are present in a system's predictions. In multi-modal settings, where inputs are both images and text, we may be able to detect objects in images that are very likely to be mentioned in the output using auxiliary image classification models (Anderson et al., 2017).

Our goal in this chapter is to find a way to force the output of a model to contain such *lexical constraints*, while still taking advantage of the distribution p_θ learned from training data. The intuition behind this goal is to *guide* the search for the best output using information that the model does not have access to, but that will result in a better output if it it is included.

As has already been discussed at length in previous chapters, real world use cases of MT are often part of a pipeline where final translations are produced by combining automatically translated output with user inputs. PE and IMT are examples of such

pipelines (Koehn, 2009a; Specia, 2011a; Foster, 2002; Barrachina et al., 2009; Green, 2014; Knowles and Koehn, 2016). Because one of the foci of this thesis is the exploration of models that can be used interactively in CAT, the experiments in this chapter are focused upon generalized lexically constrained decoding for MT models, which can also facilitate the human-guided transformation of MT output into high-quality translations.

Again, although we focus upon interactive applications for MT in our experiments, lexically constrained decoding is relevant to any scenario where a model is asked to generate a sequence $\hat{\mathbf{y}} = \{y_0 \dots y_T\}$ given both an input $\mathbf{x}$, and a set $\{\mathbf{c_0} ... \mathbf{c_n}\}$, where each $\mathbf{c_i}$ is a sub-sequence $\{c_{i0} \dots c_{ij}\}$, that must appear somewhere in $\hat{\mathbf{y}}$. This makes our work applicable to a wide range of text generation scenarios, including image description, dialog generation, abstractive summarization, and question answering.

In the following sections, we formalize the notion of lexical constraints, and propose a decoding algorithm which allows the specification of sub-sequences that are required to be present in a model's output. Individual constraints may be single tokens or multi-word phrases, and any number of constraints may be specified simultaneously.

The main contributions of this chapter are:

1. A new decoding algorithm, called *grid beam search* (GBS), that tries to find the optimal sequence which includes a set of lexical constraints,

2. A set of experiments which evaluate a simulated CAT environment where constrained decoding is incorporated into post-editing using a framework similar to pick-revise MT (Cheng et al., 2016).

3. A set of experiments evaluating the use of GBS for prediction-time domain adaptation, without requiring any retraining of a general-domain model.

4. An open source reference implementation of GBS, which supports two open-source MT decoders

The rest of the chapter is organized as follows: Section 6.1 gives the necessary background for our discussion of GBS, Section 6.2 discusses the lexically constrained decoding and the grid beam search algorithm in detail, Section 6.3 presents our experiments and analysis, and Section 6.4 discusses the key conclusions from this work.

6.1 Beam Search for Sequence Generation

Under a model parameterized by θ, let the best output sequence $\hat{\mathbf{y}}$ given input $\mathbf{x}$ be Eq. 6.1.

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \{\mathbf{y}^{[\mathbf{T}]}\}}{\operatorname{argmax}} p_\theta(\mathbf{y}|\mathbf{x}), \tag{6.1}$$

where we use $\{\mathbf{y}^{[\mathbf{T}]}\}$ to denote the set of all sequences of length T.[1] Because the number of possible sequences for such a model is $|\mathbf{v}|^T$, where $|\mathbf{v}|$ is the number of output symbols, the search for $\hat{\mathbf{y}}$ can be made more tractable by factorizing $p_\theta(\mathbf{y}|\mathbf{x})$ into Eq. 6.2:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=0}^{T} p_\theta(y_t|\mathbf{x}; \{y_0 \dots y_{t-1}\}). \tag{6.2}$$

The standard approach is thus to generate the output sequence from beginning to end, conditioning the output at each timestep upon the input $\mathbf{x}$, and the already-generated symbols $\{y_0 \dots y_{i-t}\}$. However, greedy selection of the most probable output at each timestep, i.e.:

$$\hat{y}_t = \underset{y_i \in \{\mathbf{v}\}}{\operatorname{argmax}} p(y_i|\mathbf{x}; \{y_0 \dots y_{t-1}\}), \tag{6.3}$$

risks making locally optimal decisions which are actually globally sub-optimal. On the other hand, an exhaustive exploration of the output space would require scoring $|\mathbf{v}|^T$ sequences, which is intractable for most real-world models. Thus, a search or *decoding* algorithm is often used as a compromise between these two extremes. A common solution is to use a heuristic search to attempt to find the best output efficiently (Lowerre, 1976; Pearl, 1984; Koehn, 2010a; Rush et al., 2013; Sutskever et al., 2014b). The key idea is to discard bad options early, while trying to avoid discarding candidates that may be locally risky, but could eventually result in the best overall output.

Beam search (Lowerre, 1976; Och and Ney, 2004; Sutskever et al., 2014b) is probably the most popular search algorithm for decoding sequences in NLP applications. Beam search is simple to implement, and is flexible in the sense that the semantics of the graph of beams can be adapted to take advantage of additional structure that may be available

[1] We have already motivated such models, and shown how they can be optimized, in section 2.3

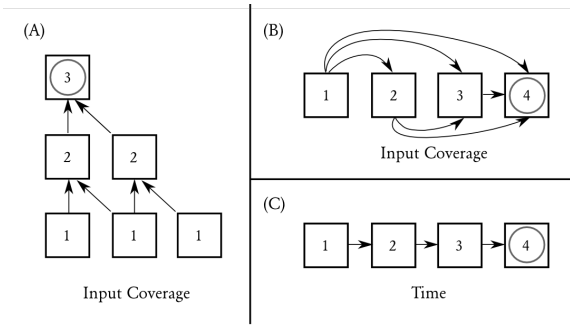


Figure 6.2: Different structures for beam search. Boxes represent beams which hold k-best lists of hypotheses. (A) Chart parsing using SCFG rules to cover spans in the input. (B) Source coverage as used in PB-SMT. (C) Sequence timesteps (as used in seq2seq models), GBS is an extension of (C). In (A) and (B), hypotheses are finished once they reach the final beam. In (C), a hypothesis is only complete if it has generated an end-of-sequence symbol.

for specific tasks. For example, in phrase-based statistical machine translation (PB-SMT) (Koehn, 2010a), beams are organized by the number of source words that are covered by the hypotheses in the beam — a hypothesis is finished when it has covered all source words. In chart-based decoding algorithms such as CYK, beams are also tied to coverage of the input, but are organized as cells in a chart, which facilitates search for the optimal latent structure of the output (Chiang, 2007). Figure 6.2 visualizes three common ways to structure search. (A) and (B) depend upon explicit structural information connecting the input and output, whereas (C) only assumes that the output is a sequence where later symbols depend upon earlier ones. Note also that (C) corresponds exactly to the bottom rows of figures 6.1 and 6.3.

From a bird's-eye view, we note that all types of heuristic search can be seen as directed graphs, where nodes are reservoirs of hypotheses sorted according to some scoring metric, and edges are operations by which a hypothesis can move from one beam to the next.

With the recent success of neural models for text generation, beam search has become the de-facto choice for decoding optimal output sequences (Sutskever et al., 2014b). However, with neural sequence models, unlike with PB-SMT, we cannot organize beams by their explicit coverage of the input. A simpler alternative is to organize beams by output timesteps from $t_0 \cdots t_N$, where N is a hyper-parameter that can be set heuristically, for example by multiplying a factor with the length of the input to make an educated guess about the maximum length of the output (Sutskever et al., 2014b). In neural text-

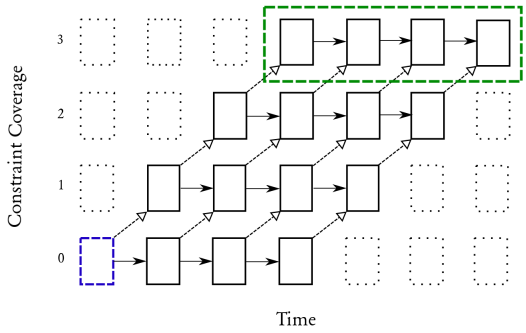


Figure 6.3: Visualizing the lexically constrained decoder's complete search graph. Each rectangle represents a beam containing k hypotheses. Dashed (diagonal) edges indicate *starting* or *continuing* constraints. Horizontal edges represent *generating* from the model's distribution. The horizontal axis covers the timesteps in the output sequence, and the vertical axis covers the constraint tokens (one row for each token in each constraint). Beams on the top level of the grid contain hypotheses which cover all constraints.

generation settings, output sequences are generally considered complete once a special EOS token has been generated. Beam size in these models is also typically kept small, and recent work has shown that the performance of some architectures can actually degrade with larger beam size (Tu et al., 2016; Koehn and Knowles, 2017), a phenomenon which is still not well-understood, and is under active investigation (Post and Vilar, 2018).

6.2 Grid Beam Search

Our goal is to organize decoding in such a way that we can constrain the search space to outputs which contain one or more pre-specified sub-sequences. We thus wish to use a model's distribution both to place lexical constraints correctly, and to generate the parts of the output which are not covered by the constraints.

Algorithm 1 presents the pseudo-code for lexically constrained decoding, see Figures 6.1 and 6.3 for visualizations of the search process. The core idea is to organize beam search as a *grid* of beams where beams are connected by typed edges that indicate the ways in which hypotheses are allowed to move from one beam to the next.

We now proceed to describe algorithm 1 in detail, as the pseudo-code alone may be somewhat difficult to understand. Beams in the search grid are indexed by t and c. The t variable tracks the timestep of the search, while the c variable indicates how many

Algorithm 1 Pseudo-code for Grid Beam Search, note that t and c indices are 0-based. Each of arguments to the the algorithm is explained in the text.

procedure CONSTRAINEDSEARCH(model, input, constraints, maxLen, numC, k)
  $\mathbf{startHyp} \Leftarrow$ model.getStartHyp(input, constraints)
  $\mathbf{Grid} \Leftarrow$ initGrid(maxLen, numC, k) ▷ initialize beams in grid
  $\mathbf{Grid}[0][0] = \mathbf{startHyp}$
  for $t = 1,\ \ t{+}{+},\ \ t < \text{maxLen}$ **do**
    for $c = \max(0, (\text{numC} + t) - \text{maxLen}),\ \ c{+}{+},\ \ c \leq \min(t, \text{numC})$ **do**
      $n, s, g = \varnothing$
      for each hyp $\in \mathbf{Grid}[t-1][c]$ **do**
        if hyp.isOpen() **then**
          $g \Leftarrow g \bigcup$ model.generate(hyp, input, constraints) ▷ generate new open hyps
        end if
      end for
      if $c > 0$ **then**
        for each hyp $\in \mathbf{Grid}[t-1][c-1]$ **do**
          if hyp.isOpen() **then**
            $n \Leftarrow n \bigcup$ model.start(hyp, input, constraints) ▷ start new constrained hyps
          else
            $s \Leftarrow s \bigcup$ model.continue(hyp, input, constraints) ▷ continue unfinished
          end if
        end for
      end if
      $\mathbf{Grid}[t][c] = \underset{h \in n \bigcup s \bigcup g}{\text{k-argmax}}$ model.score(h) ▷ k-best scoring hypotheses stay on the beam
    end for
  end for
  topLevelHyps $\Leftarrow \mathbf{Grid}[:][\text{numC}]$ ▷ get hyps in top-level beams
  finishedHyps $\Leftarrow$ hasEOS(topLevelHyps) ▷ finished hyps have generated the EOS token
  bestHyp $\Leftarrow \underset{h \in \text{finishedHyps}}{\text{argmax}}$ model.score(h)
  return bestHyp
end procedure

constraint tokens are covered by the hypotheses in the current beam[2]. Note that each step of c covers a single constraint token. In other words, constraints is an array of sequences, where individual tokens can be indexed as constraints_{ij}, i.e. token_j in constraint_i. The numC parameter in Algorithm 1 represents the total number of tokens in all constraints, which is used to set the vertical dimension of the grid.

Thus, the number of columns in our grid is equal to the number of decoding timesteps,

[2]We note that the search does not need to be organized by timestep, but for generation of natural language sequences, this will probably be the case.

and the number of rows is equal to the total number of tokens in all constraints. The k variable is the beam-size of each beam in the grid. The resulting structure actually resembles a parallelogram, instead of a rectangular grid, because the search needs to proceed at least numC timesteps before all tokens have been covered, and, if no constraints have been covered at timestep $\mathrm{maxLen} - \mathrm{numC}$, all of the constraints must be covered consecutively.

The hypotheses in a beam can be separated into two types (see lines 9-11 and 15-19 of Algorithm 1):

1. *open* hypotheses can either generate from the model's distribution, or start available constraints,

2. *closed* hypotheses can only generate the next token for a currently unfinished constraint.

At each step of the search the beam at $\mathbf{Grid}[t][c]$ is filled with candidates which may be created in three ways:

1. the *open* hypotheses in the beam to the left ($\mathbf{Grid}[t-1][c]$) may *generate* continuations from the model's distribution $p_\theta(y_i|\mathbf{x}, \{y_0 \dots y_{i-1}\})$,

2. the *open* hypotheses in the beam to the left and below ($\mathbf{Grid}[t-1][c-1]$) may *start* new constraints,

3. the *closed* hypotheses in the beam to the left and below ($\mathbf{Grid}[t-1][c-1]$) may *continue* constraints.

Therefore, the model in Algorithm 1 implements an interface with three functions: $\mathrm{generate}$, start, and $\mathrm{continue}$, which construct new hypotheses in each of the three ways. Note that the scoring function of the model does not need to be aware of the existence of constraints, but it may be, for example via a feature which indicates if a hypothesis is part of a constraint or not. We do not explore such constraint aware models in this work, but we note that pointer models (Merity et al., 2016) would be a good choice for a model that could also be *trained* to select constraints.

The beams at the top level of the grid (beams where $c = \text{numConstraints}$) contain hypotheses which cover all of the constraints. Once a hypothesis on the top level generates the EOS token, it can be added to the set of finished hypotheses. The highest scoring hypothesis in the set of finished hypotheses is the best sequence which covers all constraints.[3]

6.2.1 Multi-token Constraints

By distinguishing between open and closed hypotheses, we can allow for arbitrary multi-token phrases in the search. Thus, the set of user-provided constraints may include both individual tokens and phrases. Each hypothesis maintains a coverage vector to ensure that constraints cannot be repeated in a search path — hypotheses which have already covered constraint_i can only generate, or start constraints that have not yet been covered.

Note also that discontinuous lexical constraints, such as phrasal verbs in English or German, are easy to incorporate into GBS, by adding *filters* to the search, which require that one or more conditions must be met before a constraint can be used. For example, adding the phrasal verb "ask ⟨someone⟩ out" as a constraint would mean using "ask" as constraint_0 and "out" as constraint_1, with two filters: one requiring that constraint_1 cannot be used before constraint_0, and another requiring that there must be at least one generated token after constraint_0 before constraint_1 can be used.

6.2.2 Generalizing Prefix-Constrained Translation

We can now point out that GBS generalizes the prefix-translation usecases discussed in section 2.4 and chapter 5. In fact, to use GBS for classical IMT, it is enough to simply provide the prefix as a single constraint which begins with the Beginning-of-Sentence (BOS) token. Figure 6.4 shows the GBS search grid as it would appear in an IMT scenario: all of the constraint tokens are covered immediately, then the model continues generating according to p_θ.

The flexible structure of GBS also easily allows for more exotic usecases, such as *suffix decoding*, where the input constraint must occur *at the end* of the generated sequence. The GBS beam structure for suffix decoding is depicted in figure 6.5.

[3] Our reference implementation of GBS is available at `https://github.com/chrishokamp/constrained_decoding`

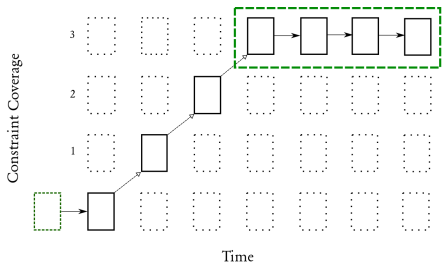


Figure 6.4: Grid beam search for IMT. First the prefix is covered, then unconstrained search continues until the EOS token is generated.

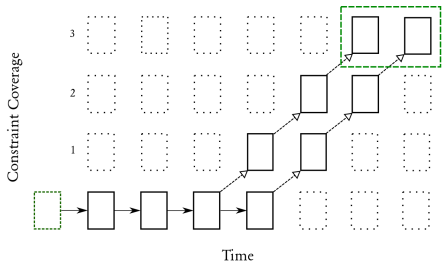


Figure 6.5: Grid beam search for suffix-based IMT. In this usecase, we know that a constraint occures at the end of the sequence, and we ask the model to generate the most likely prefix for our suffix.

6.2.3 Sub-word Units

Both the computation of the score for a hypothesis, and the granularity of the tokens (character, subword, word, etc.) are left to the underlying model. For deployment in settings where we do not control the input vocabulary, lexically constrained decoding has the same issue with OOV tokens that was discussed in the context of IMT models in chapter 5. Because our decoder can handle arbitrary constraints, there is a risk that constraints will contain tokens that were never observed in the training data, and thus are unknown by the model. Especially in domain adaptation scenarios, some user-specified constraints are very likely to contain unseen tokens. Sub-word representations provide an elegant way to circumvent this problem, by breaking unknown or rare tokens into character n-grams which are part of the model's vocabulary (Sennrich et al., 2016a; Wu et al., 2016). In the experiments in section 6.3, we use sub-word encoding as we did for the IMT models in chapter 5, ensuring that no input tokens are unknown, even if a constraint contains words which never appeared in the training data.[4]

6.2.4 Efficiency

Because the number of beams is multiplied by the number of constraints, the runtime complexity of a naive implementation of GBS is $\mathcal{O}(ktc)$. In other words, our the time complexity of our implentation of GBS is linear in the number of constraint tokens. Stan-

[4]If a *character* that was not observed in training data is observed at prediction time, it will be unknown. However, we did not observe this in any of our experiments.

dard time-based beam search is $\mathcal{O}(kt)$; therefore, some consideration must be given to the efficiency of this algorithm. Note that the beams in each column c of Figure 6.3 are independent, meaning that GBS can be parallelized to allow all beams at each timestep to be filled simultaneously. Also, we find that the most time is spent computing the states for the hypothesis candidates, so by keeping the beam size small, we can make GBS significantly faster (see chapter 7 for a discussion of a user study where users were able to interact with a constrained decoding server in real time).

Optimizing GBS

The beams in the inner **for** loop of algorithm 1 (i.e. the columns in figure 6.3) are also independent of each other, and thus can be filled in parallel, although we did not implement this in our reference implementation, because it requires us to make assumptions about the parallelization of the underlying model.

Since our work on GBS was published, a team from Amazon Research devised a way to remove the linear dependency on the number of constraint tokens from GBS (Post and Vilar, 2018). This version of GBS is a direct extension of our work, and is able to preserve all of the functionality of lexically constrained decoding, while keeping decoding time constant with respect to the number of constraints. We consider this a great enhancement of our work, and we hope it will make GBS useful in even more diverse scenarios.

We now proceed to discuss the experiments we conducted to evaluate GBS in applied scenarios, in order to confirm the practical utility of decoding algorithms which can incorporate hard lexical constraints.

6.3 Experiments

6.3.1 Models

The models used for our experiments are state-of-the-art Neural Machine Translation (NMT) systems using our own implementation of NMT with attention over the source sequence (Bahdanau et al., 2014). We used Blocks and Fuel (van Merriënboer et al., 2015) to implement our NMT models.[5] To conduct the experiments in the following sections, we trained baseline translation models for English–German (EN–DE), English–French (EN–

[5] our open source implementation is available at `https://github.com/chrishokamp/neural_imt`

FR), and English–Portuguese (EN–PT). We created a shared sub-word representation for each language pair by extracting a vocabulary of 79,000 symbols from the concatenated source and target data. See the Appendix for more details on our training data and hyper-parameter configuration for each language pair. The $\mathrm{beamSize}$ parameter is set to 10 for all experiments.

Because our experiments with lexically constrained decoding use NMT models, we can now be more explicit about the implementations of the $\mathrm{generate}$, start, and $\mathrm{continue}$ functions mentioned above for this particular GBS instantiation. For an NMT model at timestep t, $\mathrm{generate}(\mathrm{hyp}_{t-1})$ first computes a vector of output probabilities $\mathbf{o}_t = \mathrm{softmax}(g(y_{t-1}, s_i, c_i))$[6] using the state information available from hyp_{t-1}. and returns the best k continuations, i.e. Eq. 6.4:

$$\mathbf{g}_t = \underset{i}{\text{k-argmax}}\, \mathbf{o}_{ti}. \tag{6.4}$$

The start and $\mathrm{continue}$ functions simply index into the softmax output of the model, selecting specific tokens instead of doing a $\mathrm{k\text{-}argmax}$ over the entire target language vocabulary. For example, to *start* constraint $\mathbf{c}_i$, we find the score of token $\mathbf{c}_{i0}$, i.e. $\mathbf{o}_{tc_{i0}}$.

6.3.2 Pick-Revise for Interactive Post Editing

Pick-revise is an interaction cycle for MT post-editing proposed by Cheng et al. (2016). Starting with the original translation hypothesis, a (simulated) user first picks a part of the hypothesis which is incorrect, and then provides the correct translation for that portion of the output. The user-provided correction is then used as a constraint for the next decoding cycle. The pick-revise process can be repeated as many times as necessary, with a new constraint being added at each cycle.

We modify the experiments of Cheng et al. (2016) slightly, and assume that the user only provides sequences of up to three words which are missing from the hypothesis.[7] To simulate user interaction, at each iteration we chose a phrase of up to three tokens from the reference translation which does not appear in the current MT hypotheses. In the *strict* setting, the complete phrase must be missing from the hypothesis. In the *relaxed*

[6] we use the notation for the g function from Bahdanau et al. (2014)

[7] NMT models do not use explicit alignment between source and target, so we cannot use alignment information to map target phrases to source phrases

ITERATION	0	1	2	3
Strict Constraints				
EN-DE	18.44	27.64 (+9.20)	36.66 (+9.01)	**43.92** (+7.26)
EN-FR	28.07	36.71 (+8.64)	44.84 (+8.13)	**45.48** +(0.63)
EN-PT*	15.41	23.54 (+8.25)	31.14 (+7.60)	**35.89** (+4.75)
Relaxed Constraints				
EN-DE	18.44	26.43 (+7.98)	34.48 (+8.04)	**41.82** (+7.34)
EN-FR	28.07	33.8 (+5.72)	40.33 (+6.53)	**47.0** (+6.67)
EN-PT*	15.41	23.22 (+7.80)	33.82 (+10.6)	**40.75** (+6.93)

Table 6.1: Results for four simulated editing cycles using WMT test data. EN-DE uses *newstest2013*, EN-FR uses *newstest2014*, and EN-PT uses the Autodesk corpus discussed in Section 6.3.3. Improvement in BLEU score over the previous cycle is shown in parentheses. * indicates use of our test corpus created from Autodesk post-editing data.

setting, only the first word must be missing. Table 6.1 shows results for a simulated editing session with four cycles. When a three-token phrase cannot be found, we backoff to two-token phrases, then to single tokens as constraints. If a hypothesis already matches the reference, no constraints are added. By specifying a new constraint of up to three words at each cycle, an increase of close to 20 BLEU points is achieved in all language pairs.

6.3.3 Domain Adaptation via Terminology

The requirement for use of domain-specific terminologies is common in real-world applications of MT (Crego et al., 2016b). Existing approaches incorporate placeholder tokens into NMT systems, which requires modifying the pre- and post- processing of the data, and training the system with data that contains the same placeholders which occur in the test data (Crego et al., 2016b). The MT system also loses any possibility to model the tokens in the terminology, since they are represented by abstract tokens such as "⟨TERM_1⟩". An attractive alternative is to simply provide term mappings as constraints, allowing any existing system to adapt to the terminology used in a new test domain.

For the target domain data, we use the Autodesk post-editing corpus (Zhechev, 2012),

which is a dataset collected from actual MT post-editing sessions. The corpus is focused upon software localization, a domain which is likely to be very different from the WMT data used to train our general domain models. We divide the corpus into approximately 100,000 training sentences, and 1000 test segments, and automatically generate a terminology by computing the pointwise mutual information (PMI) (Church and Hanks, 1990) between source and target n-grams in the training set. We extract all n-grams from length 2–5 as terminology candidates.

$$\mathbf{pmi}(\mathbf{x};\mathbf{y}) = \mathbf{log}\frac{p(x,y)}{p(x)p(y)} \tag{6.5}$$

$$\mathbf{npmi}(\mathbf{x};\mathbf{y}) = \frac{\mathbf{pmi}(\mathbf{x};\mathbf{y})}{\boldsymbol{h}(\mathbf{x},\mathbf{y})} \tag{6.6}$$

Equations 6.5 and 6.6 show how we compute the normalized PMI for a terminology candidate pair, where $\mathbf{x}$ represents a source-language phrase, and $\mathbf{y}$ represents a target-language phrase. The PMI score is normalized to the range $[-1, +1]$ by dividing by the entropy $\boldsymbol{h}$ of the joint probability $\mathbf{p}(\mathbf{x}, \mathbf{y})$. We then filter the candidates to only include pairs whose PMI is ≥ 0.9, and where both the source and target phrases occur at least five times in the corpus. When source phrases that match the terminology are observed in the test data, the corresponding target phrase is added to the constraints for that segment. Results are shown in Table 6.2.

As a sanity check that improvements in BLEU are not merely due to the presence of the terms somewhere in the output, i.e. that the *placement* of the terms by GBS is reasonable, we also evaluate the results of randomly inserting terms into the baseline output, and of prepending terms to the baseline output.

This simple method of domain adaptation leads to a significant improvement in the BLEU score without any human intervention. Surprisingly, even an automatically created terminology combined with GBS yields performance improvements of approximately $+2$ BLEU points for En–De and En–Fr, and a gain of almost 14 points for En–Pt. The large improvement for En–Pt is probably due to the training data for this system being very different from the IT domain used in evaluation (see Appendix). Given the performance improvements from our automatically extracted terminology, manually created domain

terminologies with good coverage of the test domain are likely to lead to even greater gains. Using a terminology with GBS should be beneficial in any setting where the test domain is significantly different from the domain of the model's original training data.

System	BLEU
EN-DE	
Baseline	26.17
Random	25.18 (-0.99)
Beginning	26.44 (+0.26)
GBS	**27.99** (+1.82)
EN-FR	
Baseline	32.45
Random	31.48 (-0.97)
Beginning	34.51 (+2.05)
GBS	**35.05** (+2.59)
EN-PT	
Baseline	15.41
Random	18.26 (+2.85)
Beginning	20.43 (+5.02)
GBS	**29.15** (+13.73)

Table 6.2: BLEU Results for EN–DE, EN–FR, and EN–PT terminology experiments using the Autodesk post-editing Corpus. "Random" indicates inserting terminology constraints at random positions in the baseline translation. "Beginning" indicates prepending constraints to baseline translations.

6.3.4 Analysis

Subjective analysis of decoder output shows that phrases added as constraints are not only placed correctly within the output sequence, but also have global effects upon translation quality. This is a desirable effect for user interaction, since it implies that users can bootstrap quality by adding the most critical constraints (i.e. those that are most essential to the output), first. Table 6.3 shows several examples from the experiments in Table 6.1, where the addition of lexical constraints was able to guide our NMT systems away from initially quite low-scoring hypotheses to outputs which perfectly match the reference translations.

6.4 Discussion

This chapter has introduced lexically constrained decoding as a means of incorporating external information into the output of any model which generates natural language se-

EN-DE	
Source	
He was also an anti- smoking activist and took part in several campaigns .	
Original Hypothesis	
Es war auch ein Anti- Rauch- Aktiv- ist und nahmen an mehreren Kampagnen teil .	
Reference	**Constraints**
Ebenso setzte er sich gegen das Rauchen ein und nahm an mehreren Kampagnen teil .	(1) Ebenso setzte er
Constrained Hypothesis	(2) gegen das Rauchen
Ebenso setzte er sich **gegen das Rauchen** ein und **nahm** an mehreren Kampagnen teil .	(3) nahm
EN-FR	
Source	
At that point I was no longer afraid of him and I was able to love him .	
Original Hypothesis	
Je n'avais plus peur de lui et j'ètais capable de l'aimer .	
Reference	**Constraints**
Lá je n'ai plus eu peur de lui et j'ai pu l'aimer .	(1) Lá je n'ai
Constrained Hypothesis	(2) j'ai pu
Lá je n'ai plus **eu** peur de lui et **j'ai pu** l'aimer .	(3) eu
EN-PT	
Source	
Mo- dif- y drain- age features by selecting them individually .	
Original Hypothesis	
- Já temos as características de extracção de idade , com eles individualmente .	
Reference	**Constraints**
Modi- fique os recursos de drenagem ao selec- ion- á-los individualmente .	(1) drenagem ao selec-
Constrained Hypothesis	(2) Modi- fique os
Modi- fique os recursos de **drenagem ao selec-** ion- á-los individualmente .	(3) recursos

Table 6.3: Manual analysis of examples from lexically constrained decoding experiments. "-" followed by whitespace indicates the internal segmentation of the translation model (see Section 6.2.3)

quences. We have motivated the need for this, and invented an algorithm which can perform decoding under any set of constraints, each of which may be of any length.

Grid beam search is a flexible way to incorporate arbitrary sub-sequences into the output of any model that generates output sequences token-by-token. A wide spectrum of popular text generation models have this characteristic, and GBS should be straightforward to use with any model that already uses beam search.

In translation interfaces where translators can provide corrections to an existing hypothesis, these user inputs can be used as constraints, generating a new output each time a user fixes an error. By simulating this scenario, we have shown empirically that such a workflow can provide a large improvement in translation quality at each iteration. We showed that GBS works with state-of-the-art NMT models, and we provide an open-source implementation of GBS with integrations for our own I-NMT framework, as well as for Nematus (Sennrich et al., 2017b), a popular open-source NMT toolkit.

We have also pointed out that lexically constrained decoding generalizes all existing

frameworks for interactive machine translation, which we believe is an important theoretical contribution to the field of IMT in general.

By using a domain-specific terminology to generate target-side constraints, we have shown that a general domain model can be adapted to a new domain without any retraining. Surprisingly, this simple method can lead to significant performance gains, even when the terminology is created automatically.

In the following chapter, we show that lexically constrained decoding can be integrated into an actual CAT tool, and conduct a user evaluation in HandyCAT.

Chapter 7

Interactive Post-Editing

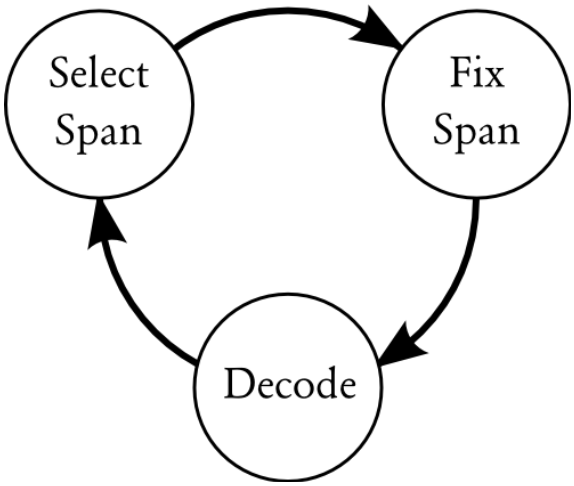


Figure 7.1: The interactive post-editing cycle

This chapter discusses a CAT component design which enables translators to take advantage of both word-level QE (as discussed in chapter 4), and constrained decoding (CD) (as introduced in Chapter 6) within an interface purpose-designed for post-editing (as prototyped in section 3.7). The work conducted in previous chapters pointed the way to a novel integration of these technologies within HandyCAT, which unifies the various domains and usecases we have studied throughout the thesis. We designed and implement such an interface, and tested the tool in a very small user study in order to evaluate the feasibility of interactive post-editing (IPE) in a realistic setting. Unfortunately, practical issues with recruitment and time meant that we only conducted a few sessions with translators using our tool; however, some insights were gained by building the IPE prototype from scratch, and testing it with actual users.

This chapter is focused upon both *RQ1* and *RQ3*, while also directly leveraging the

progress made on *RQ2* in chapter 4. The IPE prototype described below is intended to be something of a proof that we can provide some answers to all three of the high-level research questions laid out in the introduction of this thesis.

The chapter is organized as follows: section 7.1 gives an overview of our conceptualization of IPE, and the key requirements necessary to enable IPE in a CAT tool. Section 7.2 discusses the design of a new component for IPE in HandyCAT, as well as the implementation of the supporting services. Section 7.3 gives an overview of the setup of our evaluation with two professional translators. Section 7.4 presents the results of the user study, while section 7.5 covers the main conclusions of our implementation and testing of a user interface for IPE.

7.1 An Overview of Interactive Post-Editing

Although the term *interactive post-editing* has been used before (Alves et al., 2016; do Carmo, 2017)[1], to the best of our knowledge, no previous work has attempted to integrate both automatic word-level QE and interactive translation supporting arbitrary user-provided lexical constraints into a CAT tool that can be tested with human translators. Previous uses of the term IPE have either been in cases that are synonymous with IMT (Alves et al., 2016), or have referred to the interface for editing, without integration with translation and quality models supporting interactivity (do Carmo, 2017). Therefore, the term *interactive* is currently somewhat ambiguous in the context of research related to the post-editing task, so let us be specific about our requirements for an IPE interface:

In order for post-editing to be truly interactive, we believe that the following conditions must be met:

1. The user should be guided towards the portions of the translation that need editing using automatic feedback about translation quality.

2. Feedback about translation quality should be dynamic — quality annotations must be regenerated every time a change is made.

3. Translator edits should be taken as correct automatically, and always preserved when the model generates a new translation.

[1] see further background in chapter 2

4. Translators must be able to confirm portions of the translation as correct, ensuring that they will be preserved when new versions of the hypothesis are generated.

5. Revised translations can be re-generated dynamically at any time, preserving any post-editing that the user has done, as well as any spans that have been confirmed.

Our concept of IPE is most similar to the PRIMT framework introduced by Cheng et al. (2016) which was studied in chapter 6, and to segment-based IMT (Domingo et al., 2017), with the important novel distinctions that (1) we allow arbitrary reordering of constraints, (2) we use NMT instead of SMT, (3) we incorporate word-level QE, and (4) we test with real users, instead of simulating interactions.

We now proceed to define the two main roles in our interactive system, which we call the *translator* and the *model*.

The *translator* is in control of the interface, all interactions start and end based upon their explicit or implicit feedback. The *model* is a broad label for any component that has the ability to generate output automatically given a cue from the translator — in our case the model refers to the combined functionality provided by NMT with lexically constrained decoding, as well as word level QE.

As discussed in section 2.1.1, we wish to design an interface which enables these two entities to co-operate, taking advantage of the precise expert knowledge of the translator, and the fast, recall-oriented output provided by the models.

If we consider the fundamental unit of interaction during post-editing to be a *span*, which is a string of 1 or more characters, $1 \leq |\text{span}| \leq |\text{HYP}|$, where $|\text{HYP}|$ is the length of the current hypothesis, then we can define three primary operation types for IPE – *selecting* a span to edit, *fixing* one or more errors in a span, and *confirming* that a span is correct. Table 7.1 summarizes the ability of the translator and the model to perform each of these tasks.

Only *confirming* must be done by the translator alone — *selecting* and *fixing* can be performed with the aid of word-level QE and CD respectively. Our goal is to design an interface which uses QE to guide the translator during the *selection* phase, and to provide an interactive capacity to allow the translator to ask for a new translation which maintains any edits already made, as well as any spans that the translator has confirmed.

Operation	Actor(s)
Select Span	Model, Translator
Fix Span	Model, Translator
Confirm Span	Translator

Table 7.1: The three fundamental operations in IPE, along with the actor(s) (**Translator** or **Model**) who can perform these operations.

7.2 IPE Integration with CAT

Following the exploratory work conducted in section 3.7, we designed a more sophisticated version of the post-editor component, which supports the display of word-level quality information, as well as allowing integration with a server implementing lexically constrained decoding with grid beam search (hereafter referred to as CD for brevity)[2]. We note also that all components of the interface are open-source[3][4][5].

The design requirements for an IPE component are challenging to implement in practice: we need both high-performance services supporting both CD and APE-QE, as well as an intuitive interface allowing users to edit text and interact with the services naturally and reliably. The following sections discuss the design requirements and implementation of the IPE UI component and data services in turn.

7.2.1 UI Component Design

Because we use HandyCAT as our CAT platform, we can focus upon the implemention of a new target-side component with the necessary functionality (see section 3.4). Our target-area IPE component combines the Post-Editor, Word-Level QE Editor and Interactive Post-Editor components shown in Table 3.4, implementing all of the functionality of these components in a single configurable element.

Table 7.2 lists the main design requirements for our IPE component. Of these, the first is the most challenging to implement, because of the need for the text to be editable, while still rendering the quality markup, and dynamically adding and removing lexical constraint information as the translator edits.

[2]We use "CD" to refer to both the constrained decoding editing mode in the interface as discussed below, as well as to indicate the general ability of an MT model to produce translation hypotheses which incorporate arbitrary lexical constraints

[3]`https://github.com/chrishokamp/handycat`

[4]`https://github.com/chrishokamp/qe_sequence_labeling`

[5]`https://github.com/chrishokamp/constrained_decoding`

UI COMPONENT DESIGN REQUIREMENTS
• QE and user constraint information should render as colored feedback near text tokens • User should be able to select arbitrary continuous spans of text by highlighting • When a span is selected, the post-editing actions should show in a drop-down menu directly below the selected text • Component should be configurable, to allow testing of different settings

Table 7.2: Design requirements for the IPE UI component.

Name	Description	Purpose
INSERT	A cursor is placed at the end of the highlighted span, prompting the user to enter new text.	Add text that is missing from the translation hypothesis.
DELETE	Deletes the highlighted span.	Remove incorrect text.
EDIT	The highlighted span is deleted and replaced with a text cursor, prompting the user to enter new text. Whitespace is automatically added or removed around the span.	Shortcut for DELETE followed by INSERT.
CONFIRM	Marks a portion of the translation as correct. In modes which support constrained decoding, the confirmed span is guaranteed to be present when an updated translation is requested.	Allow specification of correct spans without editing.

Table 7.3: Description of the Post-Editing actions available in the IPE component.

Table 7.3 lists the editing actions available in the IPE component. INSERT and DELETE have the same semantics as in the user study in section 3.7, but operate at the *span level*, instead of at the token level. We have additionally renamed the REPLACE action to EDIT, in order to be more descriptive about its function. Finally, a CONFIRM action was added, whereby the user can assert that phrases are lexical constraints without changing the text. The purpose of the CONFIRM action is to facilitate an interactive cycle similar to PRIMT (see chapter 6), where users edit or confirm portions of the translation, then query the CD model for an updated translation which preserves these constraints. Figure 7.2 shows a screenshot of the IPE component where the user is in the process of choosing an operation to apply to a selected span.

Figure 7.2: The user has selected a span to edit, and the edit actions menu is open.

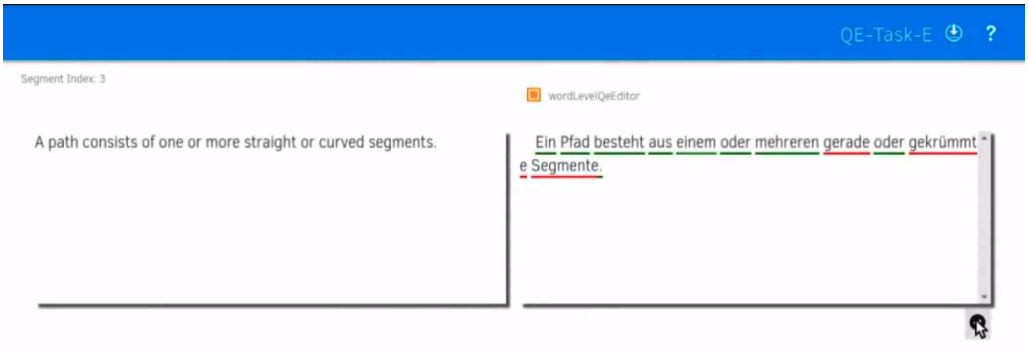


Figure 7.3: The IPE component in QE mode, green and red underlines indicate automatic quality judgements, color opacity indicates model confidence. The QE labels are automatically recomputed each time the translation is changed.

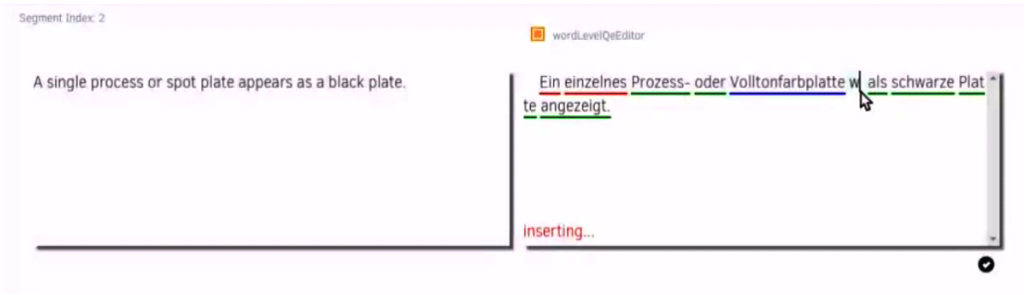


Figure 7.4: The IPE component in QE mode, user is inserting new text.

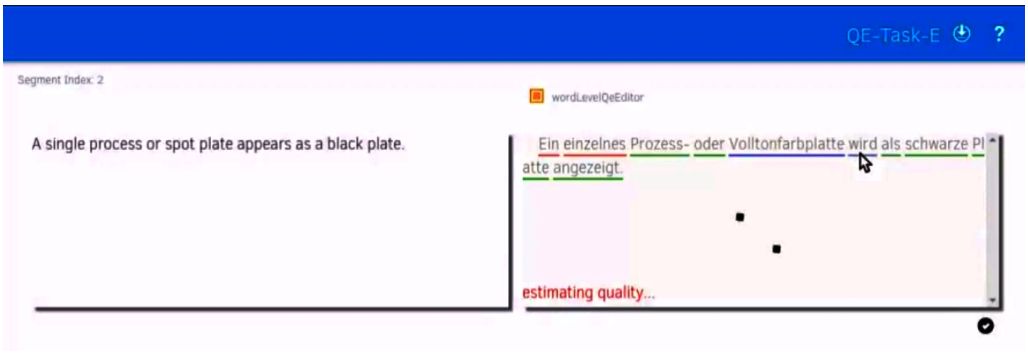


Figure 7.5: The IPE component in QE mode, quality labels are being recomputed after user has made a change

Rendering Information About Translation Quality

When the IPE component is configured to display quality information, the quality labels from the QE model are rendered as colored underlines below the tokens in the translation hypothesis. Tokens labeled as BAD are underlined in red, and tokens labeled as OK are underlined in green. Each time the translation hypothesis is changed, the quality labels are recomputed automatically. The opacity of the underline color is used to indicate the system's confidence that each QE label is correct: the lighter the color, the less confident the system is about the prediction. The way we compute QE model confidence is discussed below.

Figure 7.3 shows the IPE component in QE mode. In figure 7.4, the component is shown as the translator is in the process of making an edit to the hypothesis. Figure 7.5 shows the component with a temporary overlay which displays while QE labels are being recomputed after the hypothesis has been changed. This overlay screen displays until the server returns new labels, or until a timeout of one second has been reached. In practice, computing QE labels takes about 200 ms on average (see below).

Rendering User Constraints

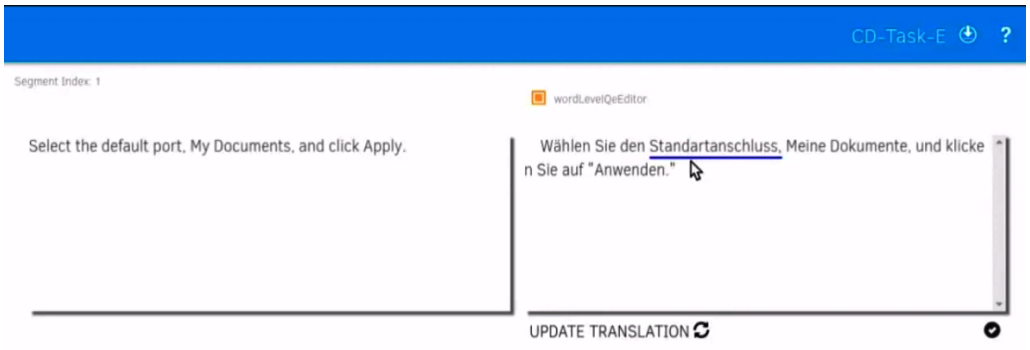


Figure 7.6: The IPE component in CD mode, after the user has inserted a constraint.

Because the IPE component will preserve any edits made by the translator when a new translation is requested, we must also provide visual feedback about which parts of the hypothesis have been contributed by the user. These "constraint spans" are underlined in blue. When the IPE component is in a mode which supports CD, a button is displayed on the bottom left of the target area, which allows the user to request an updated translation that maintains the edits and confirmed spans. Figure 7.6 shows the IPE component in CD

mode, after the user has made one edit.

When the user edits a span which is *part* of a word, such as the suffix, the component automatically extends the constraint to the entire token. This heuristic is to make sure that tokens do not get artificially separated when a new translation is generated, since we can assume that when a user edits only part of a token, they are implicitly confirming that the rest of the token is correct.

Combining QE and Constraint Information

Figure 7.7: The IPE component in IPE mode, note the presence of both QE markup, as well as the "Update Translation" button at the bottom left.

In QE and IPE modes (see below), both QE feedback and constraint feedback are rendered together. Constraint feedback always overrides QE labels, since we assume that edits provided by the translator are correct by definition. See figure 7.4, for example, where QE markup is displayed alongside a user-added constraint.

Interaction with Character Spans

In the IPE component, as in any standard CAT editing area, translators see *detokenized* text, and interact with text at the *span* level, not at the token level. Note that this is in contrast to the post-editing actions user study discussed in section 3.7, where interactions were done at the token level, and tokens were whitespace- or punctuation delimited.

Our motivation in allowing interaction with arbitrary spans is that users may wish to edit entire phrases, or just parts of words, such an incorrect affix. We also wish to expose the ability of CD to handle multiple constraints of arbitrary length. Some more details are given below about the relationship between the character level representation in the IPE interface and the token-level models supporting IPE.

SERVER DESIGN REQUIREMENTS
GENERAL • Server-side detokenization must be flawlessly aligned with UI tokenization • Server must respond to any request in less than one second
CD SERVER • Must be as fast as possible, even in the presence of many constraints • Must support constraints of any length and composition (no unknown tokens)
QE SERVER • Must align sub-word quality annotations with text spans in user interface

Table 7.4: Design requirements for servers supporting the IPE interface.

The following section discusses the design and implementation of the QE and CD services which allow translators to use the IPE component to interact with these models in real time.

7.2.2 IPE Server Design

The server requirements for the IPE component are quite demanding — we require high-quality models for both QE and CD, which are exposed to the user interface as services that can be queried in near-real time.

APE-QE Server

The APE-QE model uses the best SRC+MT model from chapter 4.3, but uses the Marian decoder (Junczys-Dowmunt et al., 2018) instead of our own NMT decoder that was used for the APE-QE experiments, so that the prediction response time is very fast. Although Marian supports both CPU and GPU based decoding, we found that decoding on GPU was essential to ensure sub-second response times. For the $(\mathrm{SRC}, \mathrm{HYP})$ pairs used in our evaluation, the APE-QE model can return a fully annotated hypothesis with confidence labels in about 200 ms, which we believe is well within the threshold of real-time usability[6].

The APE-QE server takes the current $(\mathrm{SRC}, \mathrm{HYP})$ pair from the IPE component, and concatenates the $\mathrm{SRC} + \mathrm{HYP}$ together as a single long input to the APE-QE model. Although we experimented with many model types in section 4.3, we decided to use only

[6]For a sense of the speed of annotation, please watch the walkthrough video at `https://www.youtube.com/watch?v=Abijz71Lz8Y`.

the SRC+MT model in the IPE server, instead of the full ensemble of models, in order to keep response time as low as possible.

After the APE-QE model generates an automatically post-edited pseudo-reference $\hat{\text{REF}}$ for the (SRC, HYP) pair, $\hat{\text{REF}}$ is TER-aligned with HYP to obtain the QE labels as described in chapter 4. Critically, these annotations are at the level of tokens, but the translator is interacting with character spans in the IPE component. Our method for handling this discrepancy between tokens and characters is discussed below.

Adding Confidence Information into Word-Level QE Server Responses

Drawing on insights from the work conducted in chapters 4 and 5, and because we know that predictions from our QE model are not likely to match translator's judgements exactly, we wish to give translators some graphical feedback regarding the model's confidence about each prediction. APE-QE models have the convenient property that it is trivial to produce the n-Best tag sequences, as long as the decoder implementation used by the APE-QE model supports n-best lists. In practice, we ask the APE system for the n-best automatic post-edits for the current (SRC, HYP) pair, then we obtain the QE tags for each item in the n-Best list using TER alignment. Each token in the translation hypothesis thus has n tags, each of which may be OK or BAD — we may then compute a confidence score for each token using Eq. 7.1,

$$s = \begin{cases} 1, & \text{if } \text{tag} = \text{OK} \\ 0, & \text{otherwise} \end{cases} \tag{7.1a}$$

$$\mathbf{C}_{\text{OK}} = \frac{1}{N} \sum_{1}^{N} s \tag{7.1b}$$

$$\mathbf{C}_{\text{BAD}} = 1 - \mathbf{C}_{\text{OK}} \tag{7.1c}$$

where n is the size of the n-best list from the APE-QE system[7].

In the IPE interface, each time the QE model is queried (after every change to the

[7] In our implementation, we fix the n-best list size at 10

current translation), we generate the 10-best automatic post edits for the (SRC, HYP) pair currently being edited. The QE server thus returns the most probable label for each token, as well as a confidence score that the label is correct.

CD Server

The CD server allows the user to interact with the EN–DE model described in section 6 in real-time. The CD server receives as input the source segment, as well any edits that the translator has made, which are the lexical constraints used as the CD server decodes a new translation. In the resulting new translation, the individual constraints will be preserved exactly as specified, but they may be reordered in order to produce a better translation.

As discussed in chapter 6, in our implementation of CD, the time needed to search for the best translation given constraints scales linearly with the number of constraint tokens. Therefore, there is a risk that requests containing many long constraints may take a long time to decode. We mitigate this risk by setting the beam size of the CD server to one, which does not cause a significant drop in output quality, but makes the model much faster. This server also makes use of one Titan X GPU, where decoding is done. By keeping the beam size small, and decoding on the GPU, we are able to keep the response time to under one second for all requests.

Characters, Tokens, and Spans

The QE and CD services that support the IPE component work with BPE-encoded *tokens* as the canonical units; however, translators interact with detokenized text at the character level. Therefore, we must carefully convert between the token level annotations used by the models, and the character level annotations used to render data to users in HandyCAT. Each service is responsible for detokenizing its output, and providing character-level span annotations as output. For the QE service, these annotations are character spans matching the non-whitespace tokens in the translation hypothesis. For the CD service, the annotations specify the location of the user-provided constraints after decoding. In IPE mode, when both QE and CD are available in the interface, the UI component is responsible for overriding QE annotations with user constraints. In other words, once the user has added constraints, the constraint annotation trumps the QE annotation, and constraints are always underlined in blue (see figure 7.7).

The QE and CD services together require two GPUs for sufficiently fast performance with up to 10 concurrent users. In addition we require a reasonably fast web-server to host HandyCAT itself. The hosting requirements and configuration of the tool to support IPE are thus much more demanding than the requirements of our previous user studies with HandyCAT. Nevertheless, we were able to conduct a small-scale study, which is described in the following sections.

7.3 IPE User Study Design

Having implemented and tested our proof-of-concept IPE interface, we wished to test this tool in a realistic setting with professional translators, both to obtain subjective feedback about its design, and to empirically analyze the quality of our models in a real-world setting.

Five translators (two professionals, three master's students) were recruited to participate in the user study — for two of these translators we obtained complete logs of all actions taken in the interface.[8] This section discusses the design of the user evaluation for our IPE interface.

Evaluation Datasets

We created four projects of 11 $(\mathrm{SRC}, \mathrm{HYP})$ pairs each, each with a similar distribution of sentence lengths. The segments used in all projects were selected from the WMT 2017 QE/APE development data for the English-German language pair.[9]. We chose segments from the WMT development data to ensure that (1) the evaluation data was from the same domain as the models, and (2) the models were not trained on the evaluation data, which would could lead to unrealistically good results.

Participants

The participants were recruited from Translation Studies master's degree programs at the University of Saarland, as well as from the Irish Translator's association. Despite quite significant recruiting efforts, we only obtained responses from five interested translators with expertise in the English→German language pair.

[8] Unfortunately, issues with internet connectivity caused three of the session logs to be lost, and the studies could not be repeated because of time and cost constraints

[9] recall that the APE and word level QE tasks use the same data

Setting	Description
PE	The IPE component with only the editing actions.
QE	Words are annotated with quality information. Each time the user makes a change, word level quality is recomputed.
CD	Same as PE mode, except that an additional button is available to generate a new translation. When the translation is generated, any changes made by the user are preserved as constraints.
IPE	Both Word-Level QE and Pick-Revise are enabled.

Table 7.5: The four interface configurations evaluated in the IPE user study.

The evaluation sessions were conducted remotely, in 1-hour time slots. It is worth emphasizing that the ability to conduct a user evaluation completely remotely, with no intervention or administration during the sessions is a result of our decision to implement HandyCAT as a browser-based CAT tool, which was discussed in detail in chapter 3. However, a disadvantage of implementing the compute-heavy QE and CD services as remote microservices is that the load on the services scales linearly with the number of concurrent users, and we cannot take advantage of any local processing power that may be available on a user's device. Three of the five user logs were also lost due to internet connectivity issues on the user's devices, which highlights an unanticipated risk of conducting expensive user studies in this manner.

Experiment Design

As already discussed, our implementation of the IPE component supports configuration of the functionality available to a given user in a given project. Table 7.5 lists the four settings we consider for the evaluation, which we now describe in turn:

Post-edit (PE) In the PE setting, taken as the base case for this experiment, users may only use the interactions provided by the post editing component, i.e. *editing*, *inserting*, *deleting*.

Quality estimation (QE) In the QE setting, an APE-QE system provides initial feedback in the form of word-level labels with confidence. BAD tokens are underlined in red, and OK labels are underlined in green. The opacity of each color is a function of the model's confidence as described above. Each time the hypothesis is changed, the quality

labels and confidence scores are recomputed.

Constrained decoding (CD) In the CD setting, our constrained decoding server is available to the user to re-translate the output at any time. When an edit is made, the edit is preserved as a constraint, which will be maintained across each request for a new translation. The user must explicitly request a new translation by clicking on an "Update Translation" button.

Interactive post-editing (IPE) The IPE setting allows the user to translate with constraints, but also re-computes the quality labels after any change to the hypothesis. This mode combines the functionality of the PE and CD modes.

Models

The NMT model and the APE-QE models used in this user study are respectively the best SRC and SRC+MT models trained during the APE-QE experiments described in section 4.3. We chose to use these models so that our work would be easy to extend in the future, and so the data used in the user study is publicly available. In addition, the IT domain data used in the WMT APE and QE tasks for the English–German language pair is real data that was translated by professional translators, and is likely to be similar to the type of data that translators see in the real world.

IPE Evaluation Workflow

Before beginning the translation tasks, we asked translators to watch the orientation video, and to complete a short pre-survey (results of surveys discussed below). After watching the video, the participant is free to log in to HandyCAT with their username and password, and to begin working through the projects.

Translators are required to complete one 11-segment project in each of the modes. Users complete tasks in the same order: PE, QE, CD, and IPE. However, the document order is permuted, so each user sees different documents in the different experimental settings. After completing each project, the translator was asked to submit their edit logs; however, there were several issues with this, which are discussed in section 7.4. After completing all four projects, the final step was for translators to complete a short post-survey.

7.4 Results

We begin with an overview of the general results of the evaluation sessions, then proceed to discuss the QE and GBS components separately, and finally discuss their integration in the IPE setting of the user study.

We obtained full logs of edit sessions conducted with the IPE interface from two translators, as well as five responses to pre- and post-surveys conducted before and after translators had worked through the projects in the four settings.

Pre-Survey Feedback

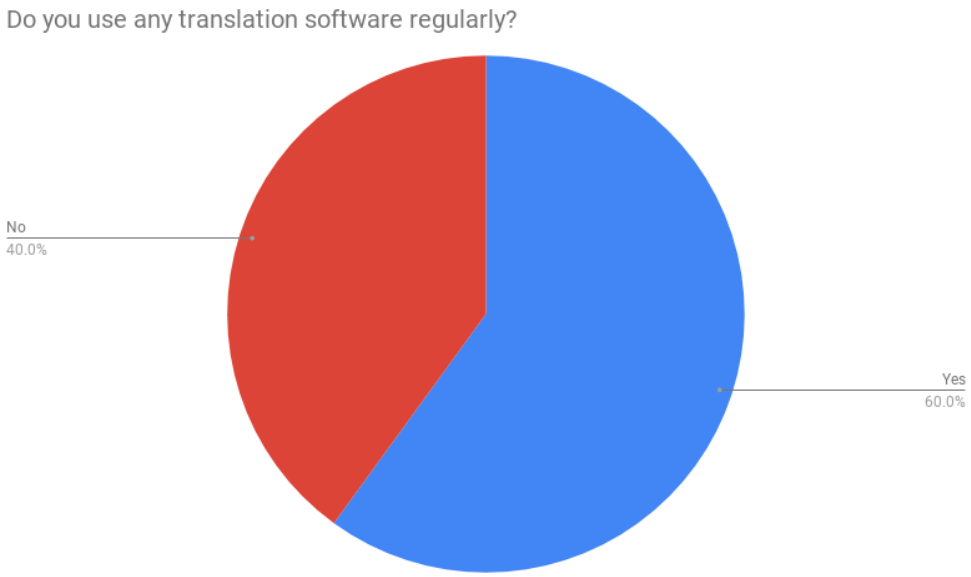


Figure 7.8: Users' experience with post-editing and translation software

All translators indicated that they had prior experience with post-editing, and three out of five regularly used translation software (Figure 7.8). All translators had at least two years of experience, and all believed that machine translation could help to make translators more productive (Figure 7.9).

Edit Log Analysis

For the two users for whom we collected complete edit logs, we conducted a deeper analysis to try to understand the effects of our design decisions upon the translation process. In the following analysis, these two users are consistently referred to as **Translator 1** and **Translator 2**. Figures 7.10 and 7.11 show the raw edit action counts taken by each of

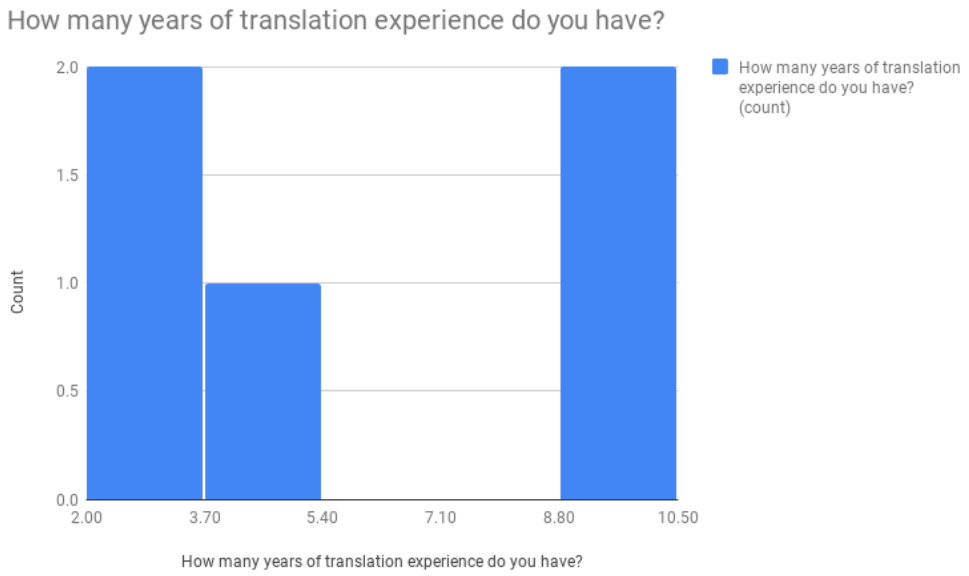


Figure 7.9: Users' years of experience in translation, beliefs about the effect of MT on translator productivity.

Pearson Corr.	p
0.67	1.87e-07

Table 7.6: Pearson correlation between per-segment edit distance for each segment for translator 1 and translator 2 (without considering the editing mode). This result shows that translators were in relative agreement about how much editing each segment needed.

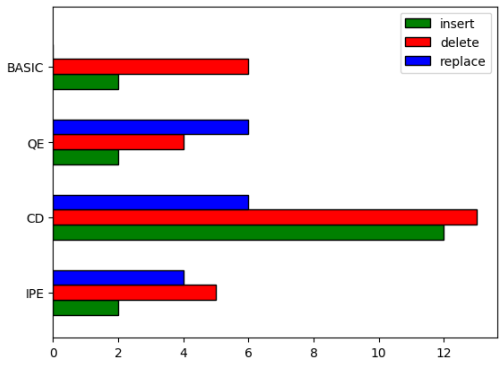


Figure 7.10: Edit actions per setting for translator 1

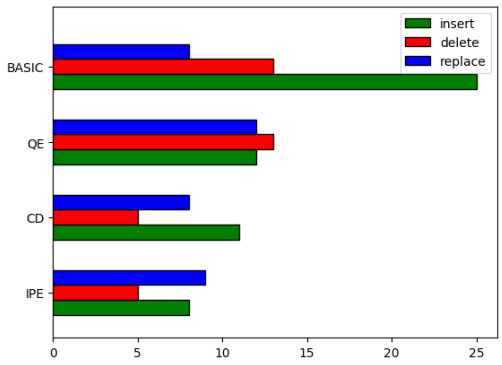


Figure 7.11: Edit actions per setting for translator 2

the translators in each of the edit modes. These figures point to significant differences between the users in their preference for the different editing actions. We notice that both translators appear to have taken fewer actions in the PE mode, which would be a positive result, but analysis reveals that this is not statistically significant, at least with our very small sample size.

Table 7.6 shows the correlation between the per-segment edit distances for Translators 1 and 2, without regard to the interface setting. This high correlation shows that users were in relative agreement about the amount of editing needed for each (SRC, HYP) pair, which is an expected result.

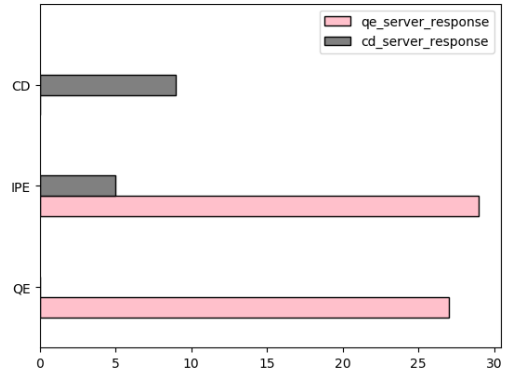


Figure 7.12: QE and CD server responses per setting for translator 1

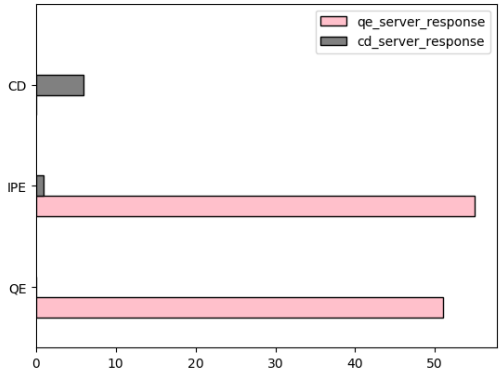


Figure 7.13: QE and CD server responses per setting for translator 2

Only a few CONFIRM events were observed across all of the evaluation sessions. We believe this is because translators did not fully understand that confirmed sub-strings were guaranteed to be present in updated translations, probably due to a lack of clarity about this functionality in the orientation video and written instructions. Likewise, in CD and IPE modes, where the additional *Update Translation* button was available, it was only used a few times by each participant. This may have been because the constrained translations returned by the model were not helpful, because users did not understand the purpose of this button, or because they simply forgot that this functionality was available. Figures 7.12 and 7.13 show the counts of server responses for the QE and CD servers. Recall that the QE server responds automatically every time the translation is updated, thus the high response counts, whereas the CD service must be explicitly queried by the translator. We observe that both translators used the CD service less in IPE mode, which was the second time they were able to interact with the service. This points to either insufficient quality of the translations, leading translators to prefer atomic editing, or to the jarring effect of large changes in the translation when this service is queried.

USER 1

QE				IPE			
TP	FP	TN	FN	TP	FP	TN	FN
118	5	6	5	102	29	12	7
$F1_{OK}$		$F1_{BAD}$		$F1_{OK}$		$F1_{BAD}$	
.96		.54		.85		.4	
$F1_{Mult}$				$F1_{Mult}$			
.52				.33			
User words edited:		11		User words edited:		41	
QE words edited:		11		QE words edited:		19	

Table 7.7: QE metrics comparing model predictions to the actual edits performed by User 1, for the QE and IPE settings.

USER 2

QE				IPE			
TP	FP	TN	FN	TP	FP	TN	FN
115	29	14	5	103	20	8	3
$F1_{OK}$		$F1_{BAD}$		$F1_{OK}$		$F1_{BAD}$	
.87		.45		.90		.41	
$F1_{Mult}$				$F1_{Mult}$			
.39				.37			
User words edited:		43		User words edited:		28	
QE words edited:		19		QE words edited:		11	

Table 7.8: QE metrics comparing model predictions to the actual edits performed by User 2, for the QE and IPE settings.

7.4.1 Evaluating the QE Model using Edit Logs

We can now perform a second evaluation on our SRC+MT word level QE model from section 4.3, by comparing its predictions with the actual edits made by translators. Tables 7.7 and 7.8 show the results of this analysis. TP, FP, TN, and FN are used to denote true positives, false positives, true negatives, and false negatives respectively. The average $F1_{mult}$ score across the settings is 0.402, which is slightly better than the baseline QE system scores in WMT 17 (Bojar et al., 2017), although these scores should not be directly compared, because the data used for the IPE user study is only a small subset of the WMT evaluation data.

In general, we find that our QE models are too conservative about tagging words as BAD, when compared with the actual edits made by translators, as evidenced by the low count of edited words for the QE models when compared to the translators, as well as the

Translator	Pearson Corr.	p
1	-0.25	0.24
2	-0.76	1.43e-05

Table 7.9: Pearson correlation between per-segment edit distance and segment-level quality approximation. Translator 1 is not in good agreement with the model, while translator 2 shows high correlation.

Question: Any general feedback on the interface?	
USER ID	RESPONSE
1	"Editing seems a bit cumbersome as the parts to be edited have to be highlighted manually rather than by double clicking on the parts to be edited. Sometimes edits weren't captured and I had to start from scratch again."
2	"There are few options to interact with the text and I think these options are well-chosen. I found that it had somewhat of a learning curve because it is not always intuitive not to be able to edit something beyond the portion of the text you want to modify."
3	"Theoretically a nice interface, although lacking options for "undo" as well as "copy/paste". Seeing a cursor would be nice and being able to use the Keyboard would be nice, as clicking and highlighting can be tedious."
4	"In general, the interface looks good but I think that the editing options can be improved to be more intuitive."
5	*no feedback given*

Table 7.10: User feedback on the IPE interface.

high false positive counts.

We can also use our word level QE tags as a proxy to sentence level QE, by getting a sentence-level quality score for each segment s_i using Eq. 7.2:

$$\text{quality}(s_i) = \frac{1}{|s_i|} \sum_{j=1}^{|s_i|} \mathbf{C}_{\text{OK}}^{j} \tag{7.2}$$

where $\mathbf{C}_{\text{OK}}^{j}$ is the model's confidence score for giving token j in s_i a label of OK, as shown in Eq. 7.1. This proxy to sentence-level quality can be evaluated by testing how well it correlates with the actual edit distance for each segment (Table 7.9). We observe a high correlation for Translator 2, but a low correlation for Translator 1, which could mean that Translator 2 trusted the QE predictions more than Translator 1 did.

Question: Any general feedback about the machine translation?	
USER ID	RESPONSE
1	"There were word order issues present, which are often happening during MT"
2	"In general it was good and I only had to change minor things."
3	"Where it was right, it was good quality. It was mostly little mistakes, with some larger ones. With tasks 3 and 4, clicking "Update Translation" after making a minor change would usually mess up the entire translation. Lacking an "undo" option, I tended not to use the "Update Translation" option."
4	"The translation quality was surprisingly good in most cases, I wonder where extreme differences in quality in some cases come from...."
5	"MT output was quite good."

Table 7.11: User feedback on the quality of the machine translation

7.4.2 Translator Feedback about the IPE Interface

In addition to the log data from two translators, we also collected general feedback from a total of five translators, asking for their opinions on many aspects of the IPE interface after they had completed a sample task in IPE mode. All translators rated the MT output as good quality. Specific feedback on the IPE interface is given in table 7.10, while feedback on the MT outputs is presented in table 7.11.

7.5 Discussion

This chapter has introduced our definition of interactive post-editing, motivated our design for a IPE integration with CAT, given an overview of our open-source implementation, and discussed the results of a very small user study conducted to test our implementation. We have shown that such an integration is feasible, and thus brought together the work conducted in chapters 3, 4, and 6. The interface worked well in the sense that users were able to successfully complete the projects within the allocated time, but the results of the benefit of QE and CD within a real CAT tool are less conclusive.

Ideally, the same or similar study could be carried out with a much larger group of subjects, in order to have sufficient samples for a deeper analysis of the effects of different $(\mathrm{SRC}, \mathrm{HYP})$ pairs, as well to further analyze the accuracy of our QE model in practice, and the places where the output of CD does not match translator's expectations. However,

a large scale user study with the IPE interface is outside of the scope of this thesis, and we leave this for future work.

Our QE models were better than random at predicting the edits that translators would make, but did not do as well as expected, given their high performance on the WMT evaluation datasets. This points to the need for multiple references when training models, or a focus upon metrics which are better at reflecting translator effort.

In this study at least, we could not confirm the usefulness of the CD server in an actual translation workflow. This is evidenced by the fact that both translator 1 and translator 2 used the CD service less in the IPE setting, having had the chance to interact with it once in the CD setting, and is also reflected in the user feedback in Table 7.10. We believe this could be due to the way this functionality was exposed, and that perhaps a more seamless design which automatically decides when to call the CD service might work better.

We believe that the correlation between our models and user behaviour might be improved by more familiarity with the interface. Our design of the IPE component was completely novel, and translators were asked to start using it rather abruptly. If some of the functionality of IPE could be integrated into a widely used CAT tool, the hypothesis that translators simply need more exposure to this new way of post-editing could be tested.

Dynamic QE and CD are difficult to integrate into CAT tools because of the real-time performance requirements of the system, and because a standardized interface between users and MT systems has not yet emerged. Thus we needed to design the UI component, servers and interfaces for IPE from scratch.

The design and engineering of the complex text interfaces required for these experiments needed a quite significant time investment. Users are interacting with spans of text in the interface, while being assisted in real-time by multiple services. Because the IPE component needs to implement such complex behaviour, and interface with multiple servers, the implementation of this component was an intricate undertaking. Although the results of this user evaluation were not as promising as we hoped, we believe that the implementation of the interface was still a worthwhile endeavor. This is, to our knowledge, the first CAT tool to use real-time constrained decoding and word-level QE to support

IPE. We believe that future work will refine and correct some of the issues with our implementation, and that components of this prototype are likely to appear in professional translation workflows quite soon.

In the end, the IPE interface received reasonably good subjective feedback from the five translators who tested it, which is perhaps the most important confirmation that this idea is worth pursuing further.

Chapter 8

Conclusion

Throughout this work we have considered both the interfaces which enable humans to transform text in one language into text in another language, and the models which can improve translation predictions, or automatic translation quality assessments, which are particularly well-suited to interactive scenarios. We have tried to improve the quality of the models themselves, and have also thought about making models better suited to interactive settings, by considering how to utilize confidence predictions, and designing ways to ensure that external lexical constraints appear in translation model output. In parallel, we developed ideas about how such models might be tested, and we built user interfaces which enable the evaluation of new designs for computer-aided translation.

The scope of this thesis has thus been rather broad, since we have considered many parts of the modern translation pipeline, from focusing on the MT models themselves in chapter 6, to designing model-intrinsic ways of improving interactivity in chapter 5, to predicting the quality of MT model output in chapter 4, to testing translation interfaces with actual users in chapters 3 and 7.

8.1 Contributions

In this section we list the core contributions of the thesis, with reference to the research goals put forth in chapter 1. The contributions can be grouped into two sets: contributions to CAT design, implementation and evaluation (Chapters 3 and 7), and contributions to machine learning models and algorithms specifically those designed for interactive translation (chapter 5 and chapter 6), and models of translation quality at the word level (chap-

ter 4).

In chapter 3, we designed and implemented HandyCAT, an open-source platform for translation process research, intending to lay the groundwork for our investigation of *RQ1*. We began by defining components, which are the basic abstract type in a CAT interface. We then introduced component-centric design, an abstract formalization of the types of components which occur in CAT interfaces. After laying out our design framework, we proceeded to show how it is implemented in HandyCAT, and covered the instantiations of CAT components that HandyCAT provides. We then introduced two pilot user studies with our tool — the first evaluated the use of autocompletion within CAT, using an SMT phrase table and language model, enabling a significant improvement in translation speed in a realistic usage scenario. The second examined a new interface for post-editing, which provides high-level atomic operations that translators use to transform MT output into human-quality translations. The user studies in this chapter motivated much of the work in the rest of the thesis, particularly the investigations into interactive NMT in chapter 5, and the interactive post-editing interface prototype designed in chapter 7.

In chapter 4, we first introduced Marmot, a framework for extracting features and training models for word level QE, developed in collaboration with Varvara Logacheva at the University of Sheffield. We then developed new models for word-level quality estimation of machine translation, and introduced a bidirectional recurrent model which, although simple, achieves good performance when compared to other single-model architectures. Finally, we showed that new formulations of QE models which use specialized features as factored inputs to sequence-to-sequence models perform better than classic sequence prediction models. This chapter provided answers to *RQ2*, by introducing novel approaches which obtain good results on word level QE tasks. Our work in this area also achieved good results in APE, although that was not one of the core research topics of this thesis. By using the APE-QE framework, we were also able to define a new way of computing the confidence of quality estimation labels, which was implemented in HandyCAT to improve user feedback on the confidence of our QE models about their predictions.

In chapter 5, we implemented an interactive version of neural machine translation, and proposed new means of leveraging confidence predictions in IMT, and methods training

confidence models for I-NMT. Chapter 5 laid the groundwork for our investigation of *RQ3*, by examining prefix-based IMT models in detail. To the best of our knowledge, this is the first work on confidence prediction for IMT since Ueffing and Ney (2005). Although the results of this work were generally underwhelming, these models may be useful starting points for future work. In addition, this work in this chapter directly inspired the invention of lexically constrained decoding and grid beam search.

In chapter 6, we introduced lexically constrained decoding with grid beam search, conceived as a means to alleviate several of the shortcomings we observed in IMT, particularly the inability to specify arbitrary lexical constraints. This chapter answers *RQ3* by confirming that a generalized formulation of interactive translation beyond prefix-based IMT is possible. Although the GBS algorithm was invented in the context of IMT, we quickly realized that this idea has impact on a much broader set of topics than MT alone, because it can be used for any model which sequentially predicts output using a factored probability distribution. We presented experiments with GBS which confirmed that it enables interactive post-editing, at least in simulated CAT scenarios, and that it can be used to domain-adapt MT models without retraining using a domain terminology at prediction time.

In chapter 7 we proposed a design for an interactive post-editing interface, which we implemented in HandyCAT. This new editing modality combines word-level QE and constrained decoding, which are both exposed as interactive models via an IPE component. We tested our design with a group of professional translators, and analyzed the edit logs from two of these translators to try to obtain more insight about the quality of the underlying models, and about our prototype interface itself. Chapter 7 attempted to unify *RQ1*, *RQ2*, and *RQ3*, by proposing an interface which relied upon concrete answers to all three of our high-level research questions to function.

8.2 Future Work

The relationship between word level QE and confidence models for IMT is an obvious synergy that we were not able to investigate in this work, and we hope to look into the potential for using QE models as proxies for confidence in the future. In addition, the

current tagging methodology for word- and phrase- level quality estimation ignores parts of the reference which are *omitted* from translation hypotheses — a serious flaw in the task definition itself. However, the current iteration of the Conference on Machine Translation (2018) does have a pilot task which includes deletion operations in the representations, which we think is a step in the right direction.

It is worth considering the possibility that MT models may soon be of such high quality that real-world use cases for QE begin to dwindle. This possibility is supported by the observation that current state-of-the-art models are at least partially based upon automatic post-editing, which simply goes around the QE labeling task, and predicts the corrected translations themselves. We believe that any researcher working on QE and related tasks should be aware of this, and should not dive too deep into task-specific models, but instead focus upon discovering the representations which are necessary and sufficient in order to know when a phrase or segment in one language entails a phrase or segment in a different language.

We also hope to evaluate lexically constrained decoding with models outside of MT, such as automatic summarization, image captioning or dialog generation. Lexically constrained decoding could also be integrated new constraint-aware models, for example via secondary attention mechanisms over lexical constraints. This is an idea that we worked on briefly during the thesis, but did not have time to explore deeply.

It would be interesting to explore the use of *structured* and *discontinuous* constraints at inference time, for example by ensuring verb agreement or phrasal verb coherence. These are already supported by our implementation of constrained decoding, but only preliminary experiments have been conducted. The possibility of structured constraints with abstract placeholders also has the potential to link decoding algorithms for NMT with the large body of work on structured decoding for SMT. We also think *negative* constraints are another interesting avenue for exploration, as a means of adding "smart" pruning to beam search and similar algorithms. Taking a step back, we believe that there is potential for further exploration into the algorithms that are currently used to extract sequences and other structures from models which factorize prediction into a series of steps. New decoding algorithms such as GBS can also be used as a means of making DL

models more interpretable, for example by studying how different models choose place certain constraints.

Finally, we hope that mainstream translation interfaces begin including technologies and designs related to our concept of interactive post-editing. In particular, lexically constrained decoding is a key feature which is needed to enable IPE designs, and is straightforward to implement and include within any modern NMT framework. We would also like to see a carefully-designed large-scale user test of an interactive post-editing tool. Since all of the component technology for this is already available, we believe the deep integration of interactive post-editing into popular CAT tools is only a matter of time.

Bibliography

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3520.

V. Alabau, R. Bonk, C. Buck, M. Carl, F. Casacuberta, González-Rubio J. Garcia Martinez, M., P. Koehn, L.A. Leiva, B. Mesa-Lao, H. Saint-Amand, C. Tsoukala, Sanchis-Trilles G., and Ortiz-Martinez D. Advanced computer aided translation with a web-based workbench. In *2nd Workshop on Post-Editing Technologies and Practice (WPTP)*, pages 53–62, Nice (France), 2 September 2013a.

V. Alabau, C. Buck, M. Carl, F. Casacuberta, M. Garcia Martinez, U. Germann, J. Gonzalez-Rubio, R. Hill, P. Koehn, L.A. Leiva, B. Mesa-Lao, D. Ortiz, H. Saint-Amand, G. Sanchis-Trilles, and C. Tsoukala. Casmacat: A computer-assisted translation workbench. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 25–28, Gothenburg (Sweden), 26 April 2014a. URL http://www.aclweb.org/anthology/E14-2007.pdf.

V. Alabau, J. González-Rubio, D. Ortíz-Martínez, F. Casacuberta, M. Garcia Martinez, B. Mesa-Lao, D.C. Petersen, B. Dragsted, and M. Carl. Integrating online and active learning in a computer-assisted translation workbench. In Francisco Casacuberta; Marcello Federico; Philipp Koehn, editor, *Workshop on Interactive and Adaptive Machine Translation. Association for Machine Translation in the Americas (AMTA)*, pages 1–8, Vancouver (Canada), 22 October 2014b. URL http://www.amtaweb.org/AMTA2014Proceedings/AMTA2014Proceedings_IAMTWorkshop_final.pdf.

Vicent Alabau, Luis Rodríguez-Ruiz, Alberto Sanchis, Pascual Martínez-Gómez, and Francisco Casacuberta. On multimodal interactive machine translation using speech recognition. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, ICMI '11, pages 129–136, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0641-6. doi: 10.1145/2070481.2070504.

Vicent Alabau, Ragnar Bonk, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Jesús González, Luis Leiva, Bartolomé Mesa-lao, Daniel Ortiz, et al. Advanced computer aided translation with a web-based workbench. In *2nd Workshop on Post-Editing Technologies and Practice*. Citeseer, 2013b.

Andrei Alexandrescu and Katrin Kirchhoff. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 1–4, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1614049.1614050.

Fabio Alves, Arlene Koglin, Bartolomé Mesa-Lao, Mercedes Garcia-Martinez, Norma

B. de Lima Fonseca, Arthur de Melo Sá, José Luiz Gonçalves, Karina Sarto Szpak, Kyoko Sekino, and Marceli Aquino. Analysing the Impact of Interactive Machine Translation on Post-editing Effort. In *New Directions in Empirical Translation Process Research: Exploring the CRITT TPR-DB*, pages 77–94. Michael Carl, Srinivas Bangalore, Moritz Schaeffer, 2016. URL https://hal.archives-ouvertes.fr/hal-01433152.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 936–945, 2017. URL https://aclanthology.info/papers/D17-1098/d17-1098.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-76297-3, 978-3-540-76297-3. URL http://dl.acm.org/citation.cfm?id=1785162.1785216.

Mahmoud Azab, Chris Hokamp, and Rada Mihalcea. Using word semantics to assist english as a second language learners. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 116–120, 2015. URL http://aclweb.org/anthology/N/N15/N15-3024.pdf.

Wilker Aziz, Sheila C. M. De Sousa, and Lucia Specia. Pet: a tool for post-editing and assessing machine translation. In *In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12*, pages 3982–3987, 2012.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Yehoshua Bar-Hillel. The present status of automatic translation of languages. *Advances in Computers*, 1:91–163, 1960.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28, March 2009. ISSN 0891-2017. doi: 10.1162/coli.2008.07-055-R2-06-29. URL http://dx.doi.org/10.1162/coli.2008.07-055-R2-06-29.

Atilim Gunes Baydin, Barak A. Pearlmutter, and Alexey Andreyevich Radul. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015. URL http://arxiv.org/abs/1502.05767.

Oliver Bender, David Vilar, Richard Zens, and Hermann Ney. Comparison of generation strategies for interactive machine translation. In *In Proceedings of EAMT 2005 (10th Annual Conference of the European Association for Machine Translation*, pages 30–40, 2005.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 1171–1179, 2015.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

Frédéric Blain, Fethi Bougares, Amir Hazem, Loïc Barrault, and Holger Schwenk. Continuous adaptation to user feedback for statistical machine translation. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1001–1005, 2015. URL http://aclweb.org/anthology/N/N15/N15-1103.pdf.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Ninth Workshop on Statistical Machine Translation*, WMT, pages 12–58, Baltimore, Maryland, 2014. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2201.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3001.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4717.

Lynne Bowker and Desmond Fisher. Computer-aided translation. In *Handbook of Translation Studies*, pages 60–65. John Benjamins Publishing Company, oct 2010. doi: 10.1075/hts.1.comp2. URL https://doi.org/10.1075/hts.1.comp2.

P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to language translation. In *Proceedings of the 12th Conference*

on Computational Linguistics - Volume 1, COLING '88, pages 71–76, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics. ISBN 963 8431 56 3. doi: 10.3115/991635.991651. URL https://doi.org/10.3115/991635.991651.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972474.

Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1596276.1596305.

Yanjun Ma Béchara, Hanna and Josef van Genabith. Statistical post-editing for a statistical mt system. In *Proc. of the MT Summit XIV*, 2011.

José Guilherme Camargo de Souza, Jesús González-Rubio, Christian Buck, Marco Turchi, and Matteo Negri. Fbk-upv-uedin participation in the wmt14 quality estimation shared-task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 322–328, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3340.

Michael Carl. A computational framework for a cognitive model of human translators. *Aslib*, 2010.

Michael Carl. Translog-ii: a program for recording user activity data for empirical reading and writing research. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Francisco Casacuberta, Jorge Civera, Elsa Cubel, Antonio L. Lagarda, Guy Lapalme, Elliott Macklovitch, and Enrique Vidal. Human interaction for high-quality machine translation. *Commun. ACM*, 52(10):135–138, October 2009. ISSN 0001-0782. doi: 10.1145/1562764.1562798. URL http://doi.acm.org/10.1145/1562764.1562798.

Francisco Casacuberta, Vicent Alabau, Jesus Gonzalez-Rubio, Luis Leiva, Daniel Ortiz-Martınez, Jorge Gonzalez Chinea, Pascual Martınez-Gomez, and Enrique Vidal Under. Interactive (-predictive) machine translation. 2014.

Asuncion Castaño and Francisco Casacuberta. A connectionist approach to machine translation. In *Fifth European Conference on Speech Communication and Technology*, 1997.

Sheila Castilho and Sharon O'Brien. Evaluating the impact of light post-editing on usability. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.

Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May 2012.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. Exploring the Planet of the APEs: a Comparative Study of State-of-the-art Methods for MT Automatic Post-Editing. In *ACL-IJCNLP 2015 : The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing : ACL2015, July 26-31, Beijing, China.- Volume 2: Short Papers*, pages 156–161, Red Hook, NY, Jul 2015. The 53rd Annual Meeting of the Association for Computational Linguistics, Beijing (Peoples R China), 26 Jul 2015 - 31 Jul 2015, Curran. doi: 10.3115/v1/P15-2026.

Yves Chauvin and David E. Rumelhart, editors. *Backpropagation: Theory, Architectures, and Applications*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995. ISBN 0-8058-1259-8.

Shanbo Cheng, Shujian Huang, Huadong Chen, Xinyu Dai, and Jiajun Chen. PRIMT: A pick-revise framework for interactive machine translation. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1240–1249, 2016. URL http://aclweb.org/anthology/N/N16/N16-1148.pdf.

A. Chesterman and E. Wagner. *Can Theory Help Translators?: A Dialogue Between the Ivory Tower and the Wordface*. Prairie Heritage. St. Jerome Pub., 2002. ISBN 9781900650496. URL https://books.google.at/books?id=vLPlAAAAMAAJ.

David Chiang. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June 2007. ISSN 0891-2017. doi: 10.1162/coli.2007.33.2.201. URL http://dx.doi.org/10.1162/coli.2007.33.2.201.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014a. URL http://aclweb.org/anthology/D/D14/D14-1179.pdf.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

Lonnie Chrisman. Learning recursive distributed representations for holistic computation. *Connection Science*, 3(4):345–366, 1991. doi: 10.1080/09540099108946592. URL https://doi.org/10.1080/09540099108946592.

Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2067–2075, 2015. URL http://jmlr.org/proceedings/papers/v37/chung15.html.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March 1990. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=89086.89095.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. Systran's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540*, 2016a.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. Systran's pure neural machine translation systems. *CoRR*, abs/1610.05540, 2016b. URL http://arxiv.org/abs/1610.05540.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 121–124, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1972-0. doi: 10.1145/2506182.2506198. URL http://doi.acm.org/10.1145/2506182.2506198.

Alex Danilo, Steve Faulkner, Arron Eicholz, Travis Leithead, and Sangwhan Moon. HTML 5.2. W3C recommendation, W3C, December 2017. https://www.w3.org/TR/2017/REC-html52-20171214/.

Aswarth Abhilash Dara, Josef van Genabith, Qun Liu, John Judge, and Antonio Toral. Active learning for post-editing based incrementally retrained MT. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 185–189, 2014. URL http://aclweb.org/anthology/E/E14/E14-4036.pdf.

Michael Denkowski, Chris Dyer, and Alon Lavie. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1370–1380, 2014. URL http://aclweb.org/anthology/P/P14/P14-1129.pdf.

Félix Emanuel Martins do Carmo. *Post-Editing: A Theoretical and Practical Challenge for Translation Studies and Machine Learning*. PhD thesis, Faculdade de Letras da Universidade do Porto, 2017.

Miguel Domingo, Alvaro Peris, and Francisco Casacuberta. Interactive-predictive translation based on multiple word-segments. *Baltic J. Modern Computing*, 4(2):282–291, 2016.

Miguel Domingo, Álvaro Peris, and Francisco Casacuberta. Segment-based interactive-predictive machine translation. *Machine Translation*, 31(4):163–185, December 2017.

ISSN 0922-6567. doi: 10.1007/s10590-017-9213-3. URL https://doi.org/10.1007/s10590-017-9213-3.

Loïc Dugast, Jean Senellart, and Philipp Koehn. Statistical post-editing on systran's rule-based translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 220–223, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626387.

Robert Eckstein, Marc Loy, and Dave Wood. *Java Swing*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1998. ISBN 1-56592-455-X.

Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel L. Forcada. Using machine translation to provide target-language edit hints in computer aided translation based on translation memories. *Journal of Artificial Intelligence Research*, 53:169–222, June 2015.

Marcello Federico, Alessandro Cattelan, and Marco Trombetti. Measuring user productivity in machine translation enhanced computer assisted translation. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, CA, October 2012. AMTA.

Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, Alberto Massidda, Holger Schwenk, Loïc Barrault, Frederic Blain, Philipp Koehn, Christian Buck, and Ulrich Germann. The matecat tool. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 129–132, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-2028.

George Foster and Guy Lapalme. Evaluation of transtype, a computer-aided translation typing system: A comparison of a theoretical- and a user- oriented evaluation procedures. In *In Conference on Language Resources and Evaluation (LREC*, 2000.

George Foster, Pierre Isabelle, and Pierre Plamondon. Target-text mediated interactive machine translation. *Machine Translation*, 12:12–175, 1997.

George F. Foster. *Text Prediction for Translators*. PhD thesis, Montreal, P.Q., Canada, Canada, 2002. AAINQ72434.

M. Garcia Martinez, K. Singla, A. Tammewar, B. Mesa-Lao, A. Thakur, M.A. Anusuya, S. Banglore, and M. Carl. Seecat: Speech & eye- tracking enabled computer assisted translation. In *European Association for Machine Translation: EAMT*, pages 81–88, Dubrovnik (Croatia), 16 June 2014. URL http://hnk.ffzg.hr/eamt2014/EAMT2014_proceedings.pdf.

William W. Gaver. Technology affordances. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 79–84, New York, NY, USA, 1991. ACM. ISBN 0-89791-383-3. doi: 10.1145/108844.108856. URL http://doi.acm.org/10.1145/108844.108856.

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedí, and Francisco Casacuberta. Interactive machine translation using hierarchical translation models. In *EMNLP*, pages 244–254. ACL, 2013. ISBN 978-1-937284-97-8.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618, 9780262035613.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. *CoRR*, abs/1301.3781, 2013.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

Spence Green. *Mixed-Initiative Natural Language Translation*. PhD thesis, Stanford, CA, United States, 2014.

Spence Green, Jeffrey Heer, and Christopher D. Manning. The efficacy of human post-editing for language translation. In *ACM Human Factors in Computing Systems (CHI)*, 2013. URL http://vis.stanford.edu/papers/post-editing.

Spence Green, Jason Chuang, Jeffrey Heer, and Christopher D. Manning. Predictive translation memory: A mixed-initiative system for human language translation. In *ACM User Interface Software & Technology (UIST)*, 2014a. URL http://idl.cs.washington.edu/papers/ptm.

Spence Green, I. Sida Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and D. Christopher Manning. Human effort and machine learnability in computer aided translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1225–1236. Association for Computational Linguistics, 2014b. URL http://aclweb.org/anthology/D14-1130.

Spence Green, Jeffery Heer, and Christopher D. Manning. Natural language translation at the intersection of ai and hci. *Queue*, 13(6), 2015. ISSN 1542-7730.

James G. Greeno. Gibson's affordances. *Psychological Review*, pages 336–342, 1994.

Gholamreza Haffari and Anoop Sarkar. Active learning for multilingual statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 181–189. Association for Computational Linguistics, 2009.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Chris Hokamp. Leveraging NLP technologies and linked open data to create better CAT tools. *Localisation Focus - The International Journal of Localisation*, 14, 2015. ISSN 1649-2358.

Chris Hokamp. Ensembling factored neural machine translation models for automatic post-editing and quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 647–654. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/W17-4775.

Chris Hokamp and Piyush Arora. Dcu-semaniacs at semeval-2016 task 1: Synthetic paragram embeddings for semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 656–662, 2016. URL http://aclweb.org/anthology/S/S16/S16-1100.pdf.

Chris Hokamp and Qun Liu. Handycat - an open-source platform for CAT tool research. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation, EAMT 2015, Antalya, Turkey, May 11 - 13, 2015*, 2015. URL https://aclanthology.info/papers/W15-4934/w15-4934.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-1141.

Chris Hokamp, Iacer Calixto, Joachim Wagner, and Jian Zhang. Target-centric features for translation quality estimation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 329–334, 2014. URL http://aclweb.org/anthology/W/W14/W14-3341.pdf.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL http://dx.doi.org/10.1016/0893-6080(89)90020-8.

Eric Horvitz. Principles of mixed-initiative user interfaces. pages 159–166. ACM Press, 1999.

Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 142–151, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL http://dl.acm.org/citation.cfm?id=2382029.2382049.

John Hutchins. The origins of the translator's workstation. *Machine Translation*, 13(4): 287–307, December 1998. doi: 10.1023/a:1008123410206. URL http://dx.doi.org/10.1023/a:1008123410206.

Pierre Isabelle, Cyril Goutte, and Michel Simard. Domain adaptation of mt systems through automatic postediting. In *Proc. of the MT Summit X*, pages 10–14, 2007.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W16-2378.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. Is neural machine translation ready for deployment? A case study on 30 translation directions. *CoRR*, abs/1610.01108, 2016. URL http://arxiv.org/abs/1610.01108.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. Is neural machine translation ready for deployment? a case study on 30 translation directions. In *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*, Seattle, WA, 2016. URL http://workshop2016.iwslt.org/downloads/IWSLT_2016_paper_4.pdf.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. *arXiv preprint arXiv:1804.00344*, 2018. URL https://arxiv.org/abs/1804.00344.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN 0131873210.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. Seattle, October 2013. Association for Computational Linguistics.

Martin Kay. The proper place of men and machines in language translation. *Machine Translation*, 12(1-2):3–23, 1997.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 329–339, 2016. URL http://aclweb.org/anthology/D/D16/D16-1032.pdf.

Rebecca Knowles and Philipp Koehn. Neural interactive translation prediction. *AMTA 2016, Vol.*, page 107, 2016.

P. Koehn. Neural Machine Translation. *ArXiv e-prints*, September 2017.

Philipp Koehn. A process study of computer-aided translation. *Machine Translation*, 23(4):241–263, 2009a. doi: 10.1007/s10590-010-9076-3. URL http://dx.doi.org/10.1007/s10590-010-9076-3.

Philipp Koehn. A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, ACLDemos '09, pages 17–20, Stroudsburg, PA, USA, 2009b. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1667872.1667877.

Philipp Koehn. A web-based interactive computer aided translation tool. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, ACLDemos '09, pages 17–20, Stroudsburg, PA, USA, 2009c. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1667872.1667877.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010a. ISBN 0521874157, 9780521874151.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010b. ISBN 0521874157, 9780521874151.

Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39. Association for Computational Linguistics, 2017. URL http://aclweb.org/anthology/W17-3204.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1557769.1557821.

Maarit Koponen, Wilker Aziz, Luciana Ramos, and Lucia Specia. Post-editing time as a measure of cognitive effort. In *AMTA 2012 Workshop on Post-Editing Technology and Practice*, WPTP, pages 11–20, San Diego, USA, 2012. URL http://www.mt-archive.info/AMTA-2012-Koponen.pdf.

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 316–322, Lisbon, Portugal, September 2015a. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3037.

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. Quality estimation from scratch (QUETCH): deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation, WMT@EMNLP 2015, 17-18 September 2015, Lisbon, Portugal*, pages 316–322, 2015b. URL http://aclweb.org/anthology/W/W15/W15-3037.pdf.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655813.

Antonio-L. Lagarda, Vicent Alabau, Francisco Casacuberta, Roberto Silva, and Enrique Díaz-de Liaño. Statistical post-editing of a rule-based machine translation system. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 217–220. Association for Computational Linguistics, 2009. URL http://www.aclweb.org/anthology/N09-2055.

Philippe Langlais, George F. Foster, and Guy Lapalme. Unit completion for a computer-aided translation typing system. *Machine Translation*, 15(4):267–294, 2000. doi: 10.1023/A:1012262211784. URL http://dx.doi.org/10.1023/A:1012262211784.

Samuel Läubli and Ulrich Germann. Statistical modelling and automatic tagging of human translation processes. *New Directions in Empirical Translation Process Research: Exploring the CRITT TPR-DB*, page 155, 2015.

Samuel Läubli, Mark Fishel, Gary Massey, Maureen Ehrensberger-Dow, and Martin Volk. Assessing post-editing efficiency in a realistic translation environment. In *Proceedings of MT Summit XIV Workshop on Post-editing Technology and Practice*, pages 83–91, 2013.

Dave Lewis, Qun Liu, Leroy Finn, Chris Hokamp, Felix Sasaki, and David Filip. Open, web-based internationalization and localization tools. *Translation Spaces, vol III*, 3(1): 99–132, 2014. doi: doi:10.1075/ts.3.05lew.

Varvara Logacheva, Chris Hokamp, and Lucia Specia. Data enhancement and selection strategies for the word-level quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 311–316, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3039.

Varvara Logacheva, Chris Hokamp, and Lucia Specia. MARMOT: A toolkit for translation quality estimation at the word level. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC*, Portorož, Slovenia, 2016. URL http://www.lrec-conf.org/proceedings/lrec2016/summaries/1054.html.

Arle Richard Lommel, Aljoscha Burchardt, and Hans Uszkoreit. Multidimensional quality metrics (mqm): A framework for declaring and describing translation quality metrics. *Tradumàtica: tecnologies de la traducció*, 0(12):455–463, 12 2014.

W. Lörscher. A Psycholinguistic Analysis of Translation Processes. *Meta*, 41(1):26–32, 1996.

Bruce T. Lowerre. *The Harpy Speech Recognition System.* PhD thesis, Pittsburgh, PA, USA, 1976. AAI7619331.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1166.

Arnt Lykke Jakobsen. *Translog documentation: version 1.0*, pages 151–186. Samfundslitteratur, Denmark, 1999. ISBN 87-593-0793-5. Opstilling: Cop.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recognition*, 45(9):3193–3203, 2012. doi: 10.1016/j.patcog.2012.01.011. URL http://dx.doi.org/10.1016/j.patcog.2012.01.011.

Pascual Martínez-Gómez, Akshay Minocha, Jin Huang, Michael Carl, Srinivas Bangalore, and Akiko Aizawa. Recognition of translator expertise using sequences of fixations and keystrokes. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pages 299–302, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2751-0. doi: 10.1145/2578153.2578201. URL http://doi.acm.org/10.1145/2578153.2578201.

André F. T. Martins, Ramón Astudillo, Chris Hokamp, and Fabio Kepler. Unbabel's participation in the WMT16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*, pages 806–811, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2387.

André Martins, Marcin Junczys-Dowmunt, Fabio Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. Pushing the limits of translation quality estimation. *Transactions of the Association for Computational Linguistics*, 5:205–218, 2017. URL https://transacl.org/ojs/index.php/tacl/article/view/1113.

Prashant Mathur, Mauro Cettolo, and Marcello Federico. Online learning approaches in computer assisted translation. In *In Proceedings of the Eight Workshop on Statistical Machine Translation*, 2013.

Warren S. McCulloch and Walter Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL http://dl.acm.org/citation.cfm?id=65669.104377.

Alan K. Melby. Multi-level translation aids in a distributed system. In *Proceedings of the 9th Conference on Computational Linguistics - Volume 1*, COLING '82, pages 215–220, Czechoslovakia, 1982. Academia Praha. doi: 10.3115/991813.991847. URL http://dx.doi.org/10.3115/991813.991847.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016. URL http://arxiv.org/abs/1609.07843.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Honza Cernocky. Rnnlm - recurrent neural network language modeling toolkit. IEEE Automatic Speech Recognition and Understanding Workshop, December 2011. URL http://research.microsoft.com/apps/pubs/default.aspx?id=175562.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-c pdf.

Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

John Moran, Christian Saam, and Dave Lewis. Towards desktop-based cat tool instrumentation. In *Third Workshop on Post-Editing Technology and Practice, The 11th Conference of the Association for Machine Translation in the Americas*, pages 99–112, 2014.

R. P. Ñeco and M. L. Forcada. Asynchronous translations with recurrent neural nets. In *Neural Networks,1997., International Conference on*, volume 4, pages 2535–2540 vol.4, Jun 1997. doi: 10.1109/ICNN.1997.614693.

Sam Newman. *Building Microservices*. O'Reilly Media, Inc., 1st edition, 2015. ISBN 1491950358, 9781491950357.

Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 681–688, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273582. URL http://doi.acm.org/10.1145/1273496.1273582.

Sharon O'Brien. Translation as human-computer interaction. *Translation Spaces*, 1(1): 101–122, 2012. URL http://doras.dcu.ie/17541/1/Translation_as_HCI_OBrien.pdf.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. of the Annual Meeting on Association for Computational Linguistics*, pages 160–167, 2003.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073133. URL http://dx.doi.org/10.3115/1073083.1073133.

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, December 2004. ISSN 0891-2017. doi: 10.1162/0891201042544884. URL http://dx.doi.org/10.1162/0891201042544884.

Daniel Ortiz-Martinez. *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation*. PhD thesis, Valencia, Spain, 2011.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. Online learning for interactive statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 546–554, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5. URL http://dl.acm.org/citation.cfm?id=1857999.1858078.

Sharon O'Brien. Towards predicting post-editing productivity. *Machine translation*, 25 (3):197–215, 2011.

Peyman Passban, Chris Hokamp, Andy Way, and Qun Liu. Improving phrase-based SMT using cross-granularity embedding similarity. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation, EAMT 2017, Riga, Latvia, May 30 - June 1, 2016*, pages 129–140, 2016. URL https://aclanthology.info/papers/W16-3403/w16-3403.

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0-201-05594-5.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

lvaro Peris, Miguel Domingo, and Francisco Casacuberta. Interactive neural machine translation. *Comput. Speech Lang.*, 45(C):201–220, September 2017. ISSN 0885-2308. doi: 10.1016/j.csl.2016.12.003. URL https://doi.org/10.1016/j.csl.2016.12.003.

Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. *CoRR*, abs/1804.06609, 2018. URL http://arxiv.org/abs/1804.06609.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL http://doi.acm.org/10.1145/2939672.2939778.

Eric Sven Ristad and Peter N. Yianilos. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532, May 1998. ISSN 0162-8828. doi: 10.1109/34.682181. URL http://dx.doi.org/10.1109/34.682181.

Raphael Rubino, Joachim Wagner, Jennifer Foster, Johann Roturier, Rasoul Samad Zadeh Kaljahi, and Fred Hollowood. DCU-Symantec at the WMT 2013 quality estimation shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 392–397, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2249.

Raphael Rubino, Tommi Pirinen, Miquel Esplà-Gomis, Nikola Ljubešić, Sergio Ortiz Rojas, Vassilis Papavassiliou, Prokopis Prokopidis, and Antonio Toral. Abu-matran at wmt 2015 translation task: Morphological segmentation and web crawling. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 184–191, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3022.

D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. E. Hinton. Schemata and sequential thought processes in PDP models. In J. L. McClelland, D. E. Rumelhart, and PDP Research Group, editors, *Parallel Distributed Processing. Volume 2: Psychological and Biological Models*, pages 7–57. MIT Press, Cambridge, MA, 1986a.

David E. Rumelhart, Geoff E. Hinton, and R. J. Wilson. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986b.

Alexander Rush, Yin-Wen Chang, and Michael Collins. Optimal beam search for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 210–221, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D13-1022.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 379–389. The Association for Computational Linguistics, 2015.

Germán Sanchis-Trilles, Daniel Ortiz-Martínez, Jorge Civera, Francisco Casacuberta, Enrique Vidal, and Hieu Hoang. Improving interactive machine translation via mouse actions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 485–494, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613776.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

Bryan Schnabel, JoAnn T. Hackos, and Rodolfo M. Raya. *A Practical Guide to XLIFF 2.0*. XML Press, 2015. ISBN 1937434141, 9781937434144.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

Mathew Stephen Seigel. *Confidence estimation for automatic speech recognition hypotheses.* PhD thesis, 2013.

Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016, August 11-12, Berlin, Germany*, pages 83–91, 2016. URL http://aclweb.org/anthology/W/W16/W16-2209.pdf.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016a. URL http://aclweb.org/anthology/P/P16/P16-1162.pdf.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016b. URL http://aclweb.org/anthology/P/P16/P16-1162.pdf.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–

68, Valencia, Spain, April 2017a. Association for Computational Linguistics. URL http://aclweb.org/anthology/E17-3017.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch-Mayne, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Miceli Barone, Jozef Mokry, and Maria Nadejde. *Nematus: a Toolkit for Neural Machine Translation*, pages 65–68. Association for Computational Linguistics (ACL), 4 2017b. ISBN 978-1-945626-34-0.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 3776–3783. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3016387.3016435.

Burr Settles. Active learning literature survey. Technical report, 2010.

Kashif Shah, Eleftherios Avramidis, Ergun Bicici, and Lucia Specia. QuEst - design, implementation and extensions of a framework for machine translation quality estimation. *The Prague Bulletin of Mathematical Linguistics*, 100:19–30, September 2013. ISSN 0032-6585. doi: 10.2478/pralin-2013-0008.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, 2007.

Jonathan Slocum. A survey of machine translation: Its history, current status, and future prospects. *Comput. Linguist.*, 11(1):1–17, January 1985. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=5615.5616.

Jason R. Smith, Herve Saint-amand, Chris Callison-burch, Magdalena Plamada, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl. In *In Proceedings of the Conference of the Association for Computational Linguistics (ACL*, 2013.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006a.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *AMTA-2006: 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA, 2006b.

Matthew G. Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. Ter-plus: Paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127, September 2009. ISSN 0922-6567. doi: 10.1007/s10590-009-9062-9. URL http://dx.doi.org/10.1007/s10590-009-9062-9.

Lucia Specia. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th Conference of the European Association for Machine Translation*, pages 73–80, 2011a.

Lucia Specia. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th Conference of the European Association for Machine Translation*, pages 73–80, 2011b.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. QuEst - a translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Sofia, Bulgaria, August 2013a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P13-4014.

Lucia Specia, Kashif Shah, Jose G. C. De Souza, Trevor Cohn, and Fondazione Bruno Kessler. Quest - a translation quality estimation framework. In *In Proceedings of the 51th Conference of the Association for Computational Linguistics (ACL), Demo Session*, 2013b.

Lucia Specia, Gustavo Paetzold, and Carolina Scarton. Multi-level translation quality prediction with quest++. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 115–120, Beijing, China, July 2015. Association for Computational Linguistics and The Asian Federation of Natural Language Processing. URL http://www.aclweb.org/anthology/P15-4020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, and Dan Tufiş. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006*, pages 2142–2147, 2006.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 14–25, 2014. URL http://aclweb.org/anthology/D/D14/D14-1003.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014a. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014b. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969173.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014c. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969173.

Carlos Teixeira and Sharon O'Brien. The impact of mt quality estimation on post-editing effort. In *In Proceedings of the MT Summit XVI, Vol. 2: Users and Translators Track*, pages 211–233, 2017.

Jörg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *LREC*, pages 2214–2218. European Language Resources Association (ELRA), 2012. ISBN 978-2-9517408-7-7.

Chase Tingley. Xliff 2.0 and the evolution of a standard. *Localisation Focus - The International Journal of Localisation*, 14, 2015. ISSN 1649-2358.

Daniel Torregrosa, Juan Antonio Pérez-Ortiz, and Mikel L Forcada. Comparative human and automatic evaluation of glass-box and black-box approaches to interactive translation prediction. *The Prague Bulletin of Mathematical Linguistics*, 108(1):97–108, 2017.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. Neural machine translation with reconstruction. *arXiv preprint arXiv:1611.01874*, 2016.

Marco Turchi, Matteo Negri, and Marcello Federico. Mt quality estimation for computer-assisted translation: Does it really help? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 530–535, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-2087.

Nicola Ueffing and Hermann Ney. Application of word-level confidence measures in interactive statistical machine translation. In *In Proceedings of EAMT 2005 (10th Annual Conference of the European Association for Machine Translation*, pages 262–270, 2005.

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation. *Comput. Linguist.*, 33(1):9–40, March 2007. ISSN 0891-2017. doi: 10.1162/coli.2007.33.1.9. URL http://dx.doi.org/10.1162/coli.2007.33.1.9.

Nancy Underwood, Bartolome Mesa-Lao, Mercedes Garcia-Martinez, Michael Carl, Vincent Alabau, Jesus González-Rubio, Luis Leiva, German Sanchis-Trilles, Daniel Ortíz-Martínez, and Francisco Casacuberta. Evaluating the effects of interactivity in a post-editing workbench. In *Language Resources and Evaluation Conference (LREC)*, pages 553–559. ACTI - International Conference Proceedings, 26 May 2014.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015. URL http://arxiv.org/abs/1506.00619.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015.

Lucas Nunes Vieira. Indices of cognitive effort in machine translation post-editing. *Machine Translation*, 28(3-4):187–216, 2014. doi: 10.1007/s10590-014-9156-x. URL https://doi.org/10.1007/s10590-014-9156-x.

Alex Waibel, Ajay N. Jain, Arthur E. McNair, Hideo Saito, Alex Hauptmann, and Joe Tebelskis. Janus: A speech-to-speech translation system using connectionist and symbolic processing strategies. In *Proceedings of the ICASSP 91*, January 1991.

Longyue Wang, Chris Hokamp, Tsuyoshi Okita, Xiaojun Zhang, and Qun Liu. The DCU discourse parser for connective, argument identification and explicit sense classification. In *Proceedings of the 19th Conference on Computational Natural Language*

Learning: Shared Task, CoNLL 2015, Beijing, China, July 30-31, 2015, pages 89–94, 2015. URL http://aclweb.org/anthology/K/K15/K15-2014.pdf.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September 2015.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1007.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings, 2015. URL http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. Max-violation perceptron and forced decoding for scalable mt training. In *EMNLP*, pages 1112–1123. ACL, 2013. ISBN 978-1-937284-97-8.

Remi Zajac. Interactive translation: a new approach, 1988.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012a. URL http://arxiv.org/abs/1212.5701.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012b.

Ventsislav Zhechev. Machine Translation Infrastructure and Post-editing Performance at Autodesk. In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*, pages 87–96, San Diego, USA, October 2012. Association for Machine Translation in the Americas (AMTA).

Appendices

Appendix A

Model Configurations

A.1 NMT System Configurations for GBS Experiments

We train all systems for 500000 iterations, with validation every 5000 steps. The best single model from validation is used in all of the experiments for a language pair. We use ℓ_2 regularization on all parameters with $\alpha = 1e^{-5}$. Dropout is used on the output layers with $p(drop) = 0.5$. We sort minibatches by source sentence length, and reshuffle training data after each epoch.

All systems use a bidirectional GRUs (Cho et al., 2014b) to create the source representation and GRUs for the decoder transition. We use AdaDelta (Zeiler, 2012a) to update gradients, and clip large gradients to 1.0.

A.1.1 English-German

Our English-German training corpus consists of 4.4 Million segments from the Europarl (Bojar et al., 2015) and CommonCrawl (Smith et al., 2013) corpora.

A.1.2 English-French

Our English-French training corpus consists of 4.9 Million segments from the Europarl and CommonCrawl corpora.

A.1.3 English-Portuguese

Our English-Portuguese training corpus consists of 28.5 Million segments from the Europarl, JRC-Aquis (Steinberger et al., 2006) and OpenSubtitles[1] corpora.

[1] http://www.opensubtitles.org/

Training Configurations	
EN-DE	
Embedding Size	300
Recurrent Layers Size	1000
Source Vocab Size	80000
Target Vocab Size	90000
Batch Size	50
EN-FR	
Embedding Size	300
Recurrent Layers Size	1000
Source Vocab Size	66000
Target Vocab Size	74000
Batch Size	40
EN-PT	
Embedding Size	200
Recurrent Layers Size	800
Source Vocab Size	60000
Target Vocab Size	74000
Batch Size	40